Air Traffic Control Area Configuration Advisories from Near-Optimal Distinct Paths

Michael Bloem[1]
NASA Ames Research Center, Moffett Field, CA, 94035

Nicholas Bambos[2]
Stanford University, Stanford, CA 94305

Area of Specialization supervisors dynamically configure available air traffic control resources so that air traffic can operate safely and efficiently. We propose to assist supervisors with this process by presenting them with a set of near-optimal and meaningfully-different configuration advisories. To find such a set of advisories, we define a problem that is equivalent to finding an optimal and other near-optimal and distinct paths in a time-expanded graph. We show that this problem is NP-hard and then motivate and specify four algorithms. One is a benchmark that solves the problem to optimality, one is a novel heuristic based on value iteration, and the third is a novel heuristic based on the A^* algorithm. The fourth algorithm solves to optimality the lowest-cost paths problem relaxation of our problem. When used to solve realistic problem instances, the lowest-cost paths algorithm rarely returned feasible solutions and the optimal algorithm required excessive computation times, but the two novel heuristics found feasible and often optimal solutions in just a few seconds. The A^*-based heuristic achieved lower computation times while the value iteration-based heuristic typically found more advisories and lower-cost advisories.

---

[1] Research Aerospace Engineer, Systems Modeling and Optimization Branch, MS 210-15. Member, AIAA. michael.bloem@nasa.gov
[2] Professor, Departments of Management Science & Engineering and Electrical Engineering. bambos@stanford.edu

Nomenclature

$\mathcal{C}$	set of all valid configuration schedules
$\mathcal{C}^{\star}_{\mathrm{CSA}}(\mathcal{C}, T)$	set of solutions to a CSA problem instance when the set of valid configuration schedules is $\mathcal{C}$ and the traffic situation data is T
$\mathcal{C}^{\varepsilon}$	set of advisories that are valid for the corresponding CSA problem instance and achieve costs within the fraction ε of the minimum cost
$\mathcal{C}_k$	set of valid configurations at configuration time step k
C	configuration schedule advisory
C_k	configuration advisory at configuration time step k
C_k^{A}	airspace configuration
C_k^{OP}	operating position configuration
C_k^{W}	workstation configuration
$\underline{C}^{\star}_{k-1}(C_k)$	previous configuration in a minimum-cost partial advisory ending at C_k
$\underline{C}^{\dagger}_{k-1}(C_k)$	previous configuration in a partial advisory ending at C_k
$\bar{C}^{\star}_{k+1}(C_k)$	next configuration in a minimum-cost partial advisory starting from C_k
$\bar{C}^{\dagger}_{k+1}(C_k)$	next configuration in a partial advisory starting from C_k
d	minimum allowable difference between returned advisories
E	the set of edges in a graph
G	a graph
g	cost
g_k	single-time step cost for a configuration at configuration time step k
g_k^{R}	reconfiguration cost for configurations at configuration time steps $k-1$ and k
g_k^{S}	static cost for a configuration at configuration time step k
I_{IS}	an instance of the Independent Set problem
$I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$	an instance of the M-ε-d-CSAs problem

$\mathcal{J}$	priority queue of configurations at configuration time steps
$J^\star$	minimum cost for a single advisory
$\underline{J}_k^\star(C_k)$	minimum cost-so-far to C_k
$\underline{J}_k(C_k)$	cost-so-far for a partial advisory ending at C_k
$\bar{J}_k^\star(C_k)$	minimum cost-to-go from C_k
$\bar{J}_k(C_k)$	cost-to-go for a partial advisory starting from C_k
$\hat{J}_k(C_k)$	under-estimate of the minimum cost-to-go from C_k
K	number of configuration time steps
k	configuration time step
M	number of requested advisories
n	largest number of valid configurations at any configuration time step
S	set of sectors
s	a sector
T	traffic situation data
T_k	traffic situation data at configuration time step k
T_t^s	Traffic situation data in sector s at traffic time step t
U	a subset of the nodes in a graph
V	the set of nodes in a graph
W	set of workstations
w	a workstation
β^{R}	reconfiguration weight
Δ	length of configuration time steps [minutes]
δ	length of traffic time steps [minutes]
ε	excess cost fraction bound

ε'	FDA^*-SC algorithm parameter
λ	parameter in FDA^* and SDA^*-SC algorithms
Φ	advisory difference metric
$\Phi_{\max}$	maximum possible advisory difference metric value
ϕ	configuration difference metric
σ	an open sector: a set of one or more sectors
$\tau(k)$	set of traffic time steps in configuration time step k

Acronyms

CSA	Configuration Schedule Advisory
FBVISAS	Forward and Backward Value Iteration with Sequential Advisory Search
FDA^*	Forward Distinct A^*
FDA^*-SC	Forward Distinct A^* with Shortcuts
ForwardVI	Forward Value Iteration
LCP	Lowest-Cost Paths
MAP	Monitor Alert Parameter
M-ε-d-CSAs	M ε-Optimal d-Distinct Configuration Schedule Advisories
OASIS	Operational Airspace Sectorization Integrated System
RecursiveVIFO	Recursive Value Iteration Fraction Optimal
ReverseVI	Reverse Value Iteration
SDA^*	Sequential Distinct A^*
SDA^*-SC	Sequential Distinct A^* with Shortcuts
VIFOEAS	Value Iteration Fraction Optimal with Exhaustive Advisory Search
ZOB	Cleveland Air Route Traffic Control Center

I. Introduction

In current air traffic management operations, a set of resources that make up an *Area of Specialization* (or just *area*) is configured by an area supervisor so that air traffic can operate safely and efficiently [1]. An area configuration specifies how airspace, air traffic controller personnel, and physical air traffic control equipment is utilized to control air traffic. Some configurations allocate these resources in a way that facilitates safe and efficient operations. For example, a safe and efficient configuration would not require a single controller to control too many aircraft at once, as this might require the controller to execute too many tasks in a short period of time. Neither would it ask a controller to control just a couple of aircraft at once, as this might make it difficult for the controller to remain engaged and attentive. Configurations are changed multiple times each day, but such changes require that additional tasks be performed by controller personnel, which may degrade the safety and efficiency of traffic operations for a period of time near the change. Although quantifying the safety and efficiency of traffic operations in an area is difficult, some algorithms have been proposed to help supervisors select area configurations [2–9]. Situations in which such algorithms could assist supervisors include the beginning and end of the night shift, when a traffic management initiative increases the volume of traffic, or when some equipment fails [10]. All of these algorithms only suggest part of the area configuration and leave the remaining parts for the supervisor to determine. For example, many algorithms do not suggest whether an open sector (i.e., a set of one or more sectors) should be allocated one or two operating positions to be staffed by controllers. Even though they are all based upon incomplete and imperfect models, none of these algorithms provide multiple advisory options for the supervisor to consider. This might be problematic because unmodeled or imperfectly-modeled aspects of area operations may make what an algorithm considers to be an optimal advisory unacceptable for deployment.

We propose to resolve these issues by presenting the supervisor with a set of diverse configuration advisories that all perform well according to an objective function. Diversity in the set of proposed advisories will increase the likelihood that one of the advisories will perform well with respect to all aspects of the area configuration problem, even those that are unmodeled or imperfectly-modeled [11]. Indeed, the results of an experiment involving retired Federal Aviation Administration

personnel demonstrate that, when presented with a set of good and diverse advisories that includes an advisory that is optimal according to an objective function, supervisors often find another advisory to be more acceptable than the optimal advisory [10]. Similar approaches have been explored in artificial intelligence (e.g., [11]), and we have provided a detailed motivation for this approach in our previous work [12]. Previous research has demonstrated that a lowest-cost path problem can be solved to find a single configuration schedule advisory that is optimal with respect to an objective function that rewards safe and efficient area operations [8, 9]. Instead of building a single advisory from an optimal path, the approach studied here builds multiple advisories by finding a set of near-optimal distinct paths.

In this article, we make five main contributions. First, we prove that finding near-optimal distinct paths to generate multiple area configuration advisories is NP-hard. Second, we propose and prove some properties of two new heuristic algorithms for this problem. One is based on value iteration and the other is based on the A^* algorithm [13]. Third, we demonstrate the inadequacy of two existing algorithms that can be applied to this problem by studying their performance on 18 small problem instances. We show that a lowest-cost paths algorithm, which solves to optimality a relaxation of the problem of interest, rarely returns feasible solutions. We also show that finding all near-optimal advisories with an algorithm proposed by Byers and Waterman [14] and then exhaustively searching through them, although guaranteed to solve the problem to optimality, requires excessive computation times. Fourth, we use these small problem instances to compare the solutions of the two novel heuristics to those of the benchmark optimal algorithm. The two heuristics typically return feasible solutions when a feasible solution exists and find optimal solutions for half of the investigated instances. Fifth, we further quantify the relative performance of the two novel heuristics by studying their performance on thousands of realistic problem instances. We find that when compared to the A^*-based heuristic, the value iteration-based heuristic offers higher-quality solutions at the expense of roughly double the computation time.

This article begins with a more detailed description of area configurations in Section II. This Section also briefly reviews the results of a human-in-the-loop experiment that involved an algorithm described in this article. Then, Section III reviews a problem that has been posed for generating a

single configuration schedule advisory and proceeds to describe the problem we studied for finding a set of advisories. Problems found in published research that are related to the multiple-advisories problem are also discussed in this Section. Four algorithms that can be applied to the multiple-advisories problem are motivated, specified, and discussed in Section IV. Next, Section V describes how we quantified the performance of the algorithms by solving problem instances based on an area in Cleveland Air Route Traffic Control Center. Finally, Section VI summarizes our contributions and main findings.

II. Background

A. Area of Specialization Configurations

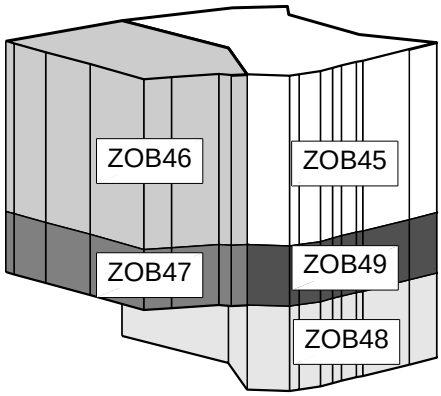
This section contains material that was also used to explain area configurations in our previous work [9, 12].

Airspace is partitioned into predefined volumes called *sectors* to facilitate the division of responsibilities between air traffic controllers. An *airspace configuration* maps a set of sectors to a set of *open sectors* such that each sector is assigned to exactly one open sector. A team of air traffic controllers, staffing one to three *operating positions*, monitors each open sector. At a minimum, a radar (also known as R-side) operating position is allocated to each open sector. A radar associate or data (also known as D-side) operating position can also be allocated to an open sector. Although rare, a third operating position can be allocated to an open sector. When more operating positions are allocated to an open sector, the tasks associated with controlling traffic in the open sector are divided among more controllers. An *operating position configuration* specifies how many operating positions are allocated to each open sector in the corresponding airspace configuration. Furthermore, each open sector is monitored from a particular *workstation* consisting of seats for air traffic controllers, a radar scope, plugs for headsets, and other equipment used by controllers to monitor traffic. Which workstation is utilized to monitor an open sector can influence how much work is involved when the open sector is changed by adding or removing sectors from it. For example, suppose an open sector consisting of two sectors has 15 aircraft in it, but that 12 of the aircraft are in one sector and 3 are in the other. Furthermore, suppose a reconfiguration is to occur in which

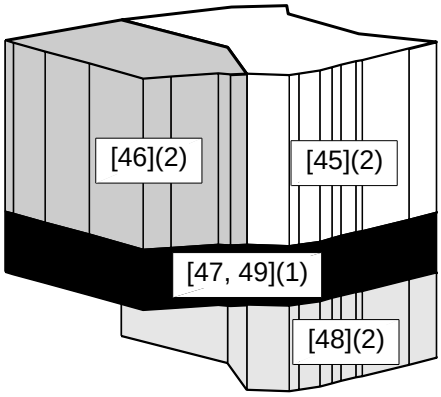
one of these sectors will be removed from the open sector and be assigned to its own open sector, operated from a different workstation. There are two choices: either transition the sector with 12 aircraft to the new open sector at the other workstation or transition the sector with 3 aircraft to the new open sector at the new workstation. Transitioning the sector with just 3 aircraft to the new open sector at the other workstation requires fewer tasks and is less disruptive. Therefore, the choice of workstation for these two open sectors can impact the safety and efficiency of corresponding air traffic operations. A *workstation configuration* specifies which workstation is utilized for monitoring each open sector in a corresponding airspace configuration. Together, a set of corresponding airspace, operating position, and workstation configurations will be referred to simply as an area *configuration*.

For example, the shapes of the five sectors in area 4 of Cleveland Air Route Traffic Control Center (ZOB) as of 20 October 2011 are shown in Fig. 1(a). The shapes of the open sectors in a sample airspace configuration are shown in Fig. 1(b) and the floor layout of corresponding operating position and workstation configurations is shown in Fig. 1(c). The airspace configuration contains four open sectors. Three of these open sectors consist of airspace from only a single sector (ZOB45, ZOB46, and ZOB48). These three open sectors are each allocated two operating positions (indicated by the number in parentheses in Figs. 1(b) and 1(c)). The fourth open sector consists of the combined airspace of sectors ZOB47 and ZOB49 and is controlled by a single operating position. In Fig. 1(c), the two workstations on the left side are used by the four operating positions allocated to the open sectors consisting of ZOB45 and ZOB46. The workstation at the top of the right side is used by the R- and D-side operating positions allocated to the open sector consisting of ZOB48. Finally, the single R-side operating position controlling the open sector consisting of ZOB47 and ZOB49 is using the bottom workstation on the right side.

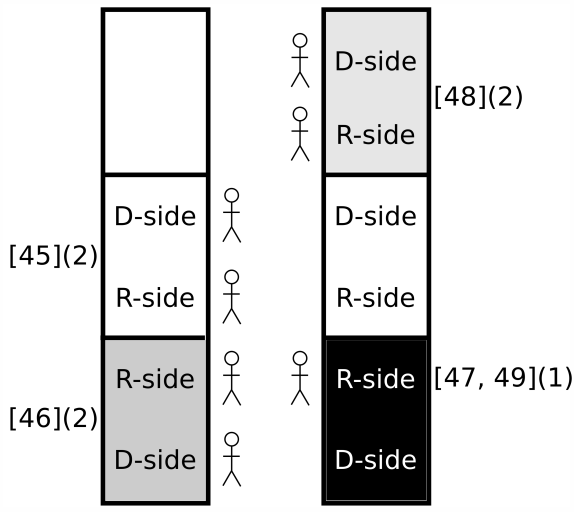
This specification of area configurations is incomplete. The main missing component is a mapping of available controllers to operating positions. This component of configurations is excluded from the problem statement because factors that influence this component, such as controller skill, fatigue, and personality, may be difficult to quantify. This component of the configuration is left for the supervisor to determine without the assistance of an advisory. However, by providing multiple



(a) Sectors.



(b) Sample airspace configuration.



(c) Sample operating position and workstation configurations.

Fig. 1 Sectors and sample configuration of ZOB area 4.

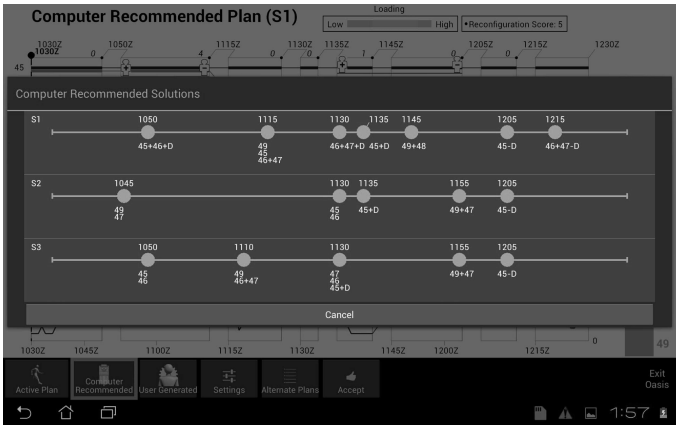


Fig. 2 Screenshot of the OASIS decision-support tool.

configuration advisories, instead of just a single advisory, we hope to increase the likelihood that the supervisor will be presented with an option that performs well enough with respect to this unmodeled aspect of configurations as well as other unmodeled or imperfectly-modeled aspects.

B. Human-in-the-Loop Experiment Results

Lee et al. conducted a part-task human-in-the-loop experiment in which the Sequential Distinct A^* with Shortcuts algorithm described in sub-section IV.C.4 was utilized by a decision-support tool referred to as the Operational Airspace Sectorization Integrated System (OASIS) [10]. Eight retired Federal Aviation Administration personnel each used the tool in four simulated scenarios. Figure 2 shows a screenshot of the OASIS tool, an application that runs on an Android touch tablet, in which the user is presented with three advisories. Results of this experiment suggest that presenting multiple near-optimal and distinct advisories added value because in more than 60% of the cases in which the algorithm presented users with more than one advisory, users found the second or third advisory to be more acceptable than the first advisory, even though the first advisory was optimal according to the objective function we used to evaluate the safety and efficiency of operations (see sub-section III.A.4). Furthermore, when asked how many advisories they wanted the tool to provide, participants that worked with this algorithm requested an average of 2.8 advisories.

III. System Modeling and Problem Formulation

The problem we studied for finding multiple configuration schedule advisories is based upon the approach for finding a single configuration schedule advisory proposed and motivated in [8, 9]; it will be summarized in sub-section III.A for completeness and to introduce notation. We recently motivated and posed the multiple-advisory problem studied here [12]; this problem will be described in sub-section III.B and proven to be NP-hard in Appendix A. Sub-section III.C describes related research and positions the multiple-advisories problem relative to other problems that involve finding multiple paths.

A. Configuration Schedule Advisory Problem

The Configuration Schedule Advisory (CSA) problem specifies a lowest-cost path problem on a time-extended graph. The solution of this problem corresponds to an area configuration schedule advisory.

1. Decision Variables.

The time horizon of the schedule is broken into $K+1$ discrete configuration time steps $k = 0, 1, 2, \ldots, K$ of length Δ minutes. The configuration time step $k = 0$ is used for data describing the state of the area at the time an advisory is generated.

The decision variables that make up a configuration schedule advisory C are C_k for $k \in \{0, 1, 2, \ldots, K\}$, where C_k is the advised configuration at configuration time step k. More concretely, a configuration schedule advisory for configuration time step k is $C_k = \{C_k^{\mathrm{A}}, C_k^{\mathrm{OP}}, C_k^{\mathrm{W}}\}$ and it consists of an airspace configuration C_k^{A}, a corresponding operating position configuration C_k^{OP}, and a corresponding workstation configuration C_k^{W}. For a given set of airspace sectors $S = \{s_1, s_2, \ldots, s_{|S|}\}$ under consideration, an airspace configuration consists of a set of open sectors $C_k^{\mathrm{A}} = \{\sigma_1, \sigma_2, \ldots, \sigma_{|C_k^{\mathrm{A}}|}\}$. Each open sector $\sigma \in C_k^{\mathrm{A}}$ is itself a set consisting of at least one sector from S. An operating position configuration C_k^{OP} is a mapping that specifies whether one or two operating positions are allocated to each open sector in the corresponding airspace configuration. The formulation can be extended to allow the allocation of three operating positions to an open sector, but in this article we only allow the typical values of one or two operation positions per open

sector. Finally, a workstation configuration C_k^{W} is a mapping from open sectors in C_k^{A} to the set of available workstations W.

2. Data.

The traffic situation data T is a set consisting of a data element T_k for each configuration time step $k \in \{0, 1, \ldots, K\}$. Furthermore, each T_k contains the traffic situation data for each sector during configuration time step k: $T_k = \{T_k^{s_1}, T_k^{s_2}, \ldots, T_k^{s_{|S|}}\}$. Generally, this traffic situation data must contain any predicted air traffic data required to compute the problem objective function. Although many other objective function formulations are possible, the function specified in subsection III.A.4 for use in this article requires that T_k^s contain a unique identifier for each flight in sector s at each *traffic time step* during configuration time step k. Since air traffic characteristics often change faster than airspace configurations, we further discretize time into traffic time steps of length δ minutes (where $\delta \leq \Delta$). Let $\tau(k)$ be the set of traffic time steps in configuration time step k. Then each T_k^s is itself a set containing the traffic situation data in sector s at each traffic time step $t \in \tau(k)$: $T_k^s = \{T_t^s\}_{t \in \tau(k)}$. Finally, each T_t^s contains a unique identifier for each aircraft located within s during t.

Other data required for the objective function used in this article are the capacities of open sectors. The capacity of an open sector is the maximum number of aircraft that can safely be within the open sector simultaneously when the open sector is allocated two operating positions. An open sector *Monitor Alert Parameter* (*MAP*) is used as a capacity bound in current air traffic operations and so MAP values are used as the required sector capacity data.

3. Constraints.

A configuration schedule advisory C must be in the set $\mathcal{C}$ of all valid configuration schedules. Although $\mathcal{C}$ could be defined more generally, here it is specified as a set of valid configurations at each configuration time step: $\mathcal{C} = \{\mathcal{C}_k\}_{k=0}^{K}$. The set $\mathcal{C}_0$ contains only the current configuration C_0. We typically utilize historical data to help construct $\mathcal{C}$, but there may be valid configurations that could be included in $\mathcal{C}$ even if they have not been used historically. To ensure that $\mathcal{C}$ is appropriate, subject-matter experts should be consulted.

Valid configurations in $\mathcal{C}_k$ must fulfill several fundamental requirements that apply to any problem instance and any configuration time step. For example, open sectors must be spatially contiguous. Airspace configurations at each configuration time step must assign each sector to exactly one open sector. Only one or two operating positions can be allocated to any open sector. Each open sector must be assigned to a single workstation, and a workstation cannot be assigned to multiple open sectors.

Valid configurations can also be specific to certain problem instances and may apply for all or only a subset of configuration time steps. For example, configurations containing certain open sectors might be denoted as invalid because they are geographically too large to be displayed clearly on a scope. Other configurations might be invalid for some period of time due to temporary workstation equipment outages. More permanent technological limitations, such as radio frequency coverage issues, may also limit the set of valid configurations. Training sessions may require that certain open sectors be a part of any configuration utilized for certain configuration time steps. The number of available controller personnel can impose an upper bound on the number of operating positions that can be used in a configuration. This list is not exhaustive: any configuration can be removed from consideration during any configuration time step.

4. Objective.

The problem objective is to minimize a configuration schedule cost $g(C, T)$. The cost for a configuration schedule advisory is a sum of the costs incurred by the scheduled configuration at each configuration time step in the time horizon:

$$g(C, T) = \sum_{k=1}^{K} g_k(C_{k-1}, T_{k-1}, C_k, T_k). \tag{1}$$

For a single configuration time step, the cost is a weighted sum of a *static cost* and a *reconfiguration cost*:

$$g_k(C_{k-1}, T_{k-1}, C_k, T_k) = g_k^{\mathrm{S}}(C_k, T_k) + \beta^{\mathrm{R}} g_k^{\mathrm{R}}(C_{k-1}, T_{k-1}, C_k, T_k), \tag{2}$$

where β^{R} is the *reconfiguration weight*. The static cost penalizes configurations that do not facilitate safe and efficient operations during the configuration time step, such as those that require

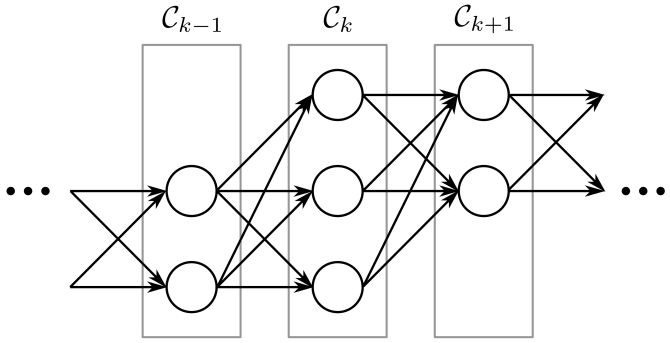


Fig. 3 Portion of the graph for a sample CSA problem instance.

controllers to control excessively high or low amounts of traffic. The reconfiguration cost penalizes changes in configurations that occur between configuration time steps, particularly those that require controllers to transfer control of many aircraft from one workstation to another. These cost functions are complex and involve many parameters; a detailed specification of these costs and suggested parameter values can be found in [8] and [9]. The reconfiguration weight determines the relative importance of these two competing types of cost, and [8] and [9] found suggested values for it by studying historical configuration and air traffic data.

5. Problem Statement Summary.

The CSA problem is

$$\text{minimize } g(C,T) \tag{3}$$

$$\text{subject to } C_k \in \mathcal{C}_k, \quad k = 0,1,2,\ldots,K. \tag{4}$$

The CSA problem can be mapped to a lowest-cost path problem on a time-expanded graph [8, 9]. A sample portion of such a time-expanded graph is depicted in Fig. 3. Each valid configuration at a time step is a node in the graph and transitions between configurations in one time step to configurations in the next time step are edges. Node costs correspond to the static cost and edge costs correspond to the reconfiguration cost. The origin for the path is dictated by the initial configuration C_0. Any configuration in $\mathcal{C}_K$ is a valid destination for the path.

For certain types of optimization problems (such as linear programs), instances with more constraints may require algorithms to perform more computations before finding a solution. This

is not the case for the CSA problem. For CSA problem instances, additional constraints mean that more configurations (nodes) must be removed from the relevant time-expanded graph. This does add some complexity to the process of specifying the graph for a problem instance, but specifying the instance is typically much less computationally expensive than solving it. Furthermore, the computational burden of finding a lowest-cost path decreases as graph size decreases (i.e., as nodes and edges are removed), so additional constraints lead to CSA problem instances that require fewer computations to solve.

B. M ε-Optimal d-Distinct Configuration Schedule Advisories Problem

In this sub-section, we extend the CSA problem to require that the solution consists of a set of M valid advisories. Former and current area supervisors requested that a minimum-cost advisory always be returned, so the first advisory must be optimal for the corresponding CSA problem. Each other returned advisory must achieve a cost value that is within some fraction of the minimum cost value. This constraint is intended to ensure that supervisors do not waste time evaluating advisories that are unlikely to be useful because they perform significantly sub-optimally with respect to aspects of the problem captured by the objective function. Finally, the advisories in each pair of returned advisories must be sufficiently different from each other according to an advisory difference metric. The purpose of this constraint is to encourage the returned advisories to perform differently with respect to unmodeled or imperfectly-modeled components of the area configuration problem faced by supervisors. Such diversity of performance increases the chance that at least one advisory will be feasible with respect to aspects of the problem that are unmodeled or imperfectly-modeled [11]. Our previous work provides further motivation for this problem statement [12]. This problem statement is also motivated by the results of a human-in-the-loop experiment in which retired Federal Aviation Administration personnel were presented with advisories from an algorithm that attempts to solve this problem (see sub-section II.B and [10]).

The M ε-Optimal d-Distinct Configuration Schedule Advisories (M-ε-d-CSAs) problem is

$$\text{minimize} \quad \sum_{m=1}^{M} g(C^m, T) \tag{5}$$

$$\text{subject to } |\mathcal{C}^M| = M \tag{6}$$

$$C_k^m \in \mathcal{C}_k, \quad k = 0, 1, 2, \ldots, K, \ m = 1, 2, \ldots, M \tag{7}$$

$$C^1 \in \mathcal{C}^\star_{\text{CSA}}(\mathcal{C}, T) \tag{8}$$

$$\frac{g(C^m, T) - g(C^1, T)}{g(C^1, T)} \leq \varepsilon \quad m = 2, 3, \ldots, M \tag{9}$$

$$\Phi(C^m, C^{m'}) \geq d \quad \forall m, m' \in \{1, 2, \ldots, M\}, \ m \neq m'. \tag{10}$$

Here $\mathcal{C}^M = \{C^1, \ldots, C^M\}$ is the set of $M \in \mathbb{Z}_{++}$ advisories that make up a solution to the problem. The objective (5) is to minimize the sum of the costs of the M advisories. Constraint (6) requires that M advisories be returned. Although not required by the problem statement, in the event that M feasible advisories cannot be found, the algorithms we devise and investigate will return a set containing fewer advisories. Constraint (7) ensures that each advisory is valid. The set $\mathcal{C}^\star_{\text{CSA}}(\mathcal{C}, T) \subseteq \mathcal{C}$ is the set of solutions to a CSA problem instance when the set of valid configuration schedules is $\mathcal{C}$ and the traffic situation data is T. Therefore, constraint (8) requires that the first advisory be an optimal advisory for the corresponding CSA problem. Constraint (9) requires that each other advisory achieves a cost that does not exceed a particular value. More precisely, the cost of each other advisory in excess of the minimum advisory cost ($g(C^m, T) - g(C^1, T)$), expressed as a fraction of the minimum advisory cost $g(C^1, T)$, must not exceed an *excess cost fraction bound* $\varepsilon \in \mathbb{R}_+$. Finally, constraint (10) requires that each pair of advisories, when compared using an advisory difference metric $\Phi : \mathcal{C} \times \mathcal{C} \to \mathbb{R}_+$, achieve a difference of at least $d \in \mathbb{R}_+$.

The M-ε-d-CSAs problem is NP-hard. This is shown in Appendix A by demonstrating that the Independent Set (IS) problem, which is NP-complete and even difficult to approximate [15], is polynomial-time reducible to M-ε-d-CSAs. This is sufficient to show that the M-ε-d-CSAs problem is NP-hard [16].

1. Advisory Difference Metric.

The advisory difference metric Φ maps a pair of advisories from $\mathcal{C}$ to a non-negative real number. It could take many forms, but in this article it is defined as

$$\Phi(C, C') = \sum_{k=1}^{K} \phi(C_k, C'_k), \tag{11}$$

where $\phi : \mathcal{C}_k \times \mathcal{C}'_k \to \mathbb{R}_+$ is a configuration difference metric. It defines a difference between valid configurations.

For this article, the configuration difference metric is

$$\phi(C_k, C'_k) = \begin{cases} 1 & \text{if } C_k^{\mathrm{A}} \neq C_k'^{\mathrm{A}} \\ 0 & \text{else.} \end{cases} \tag{12}$$

Pairs of configurations that use different airspace configurations (sets of open sectors) achieve a configuration difference of 1, while all other pairs of configurations achieve a configuration difference of 0. Therefore, when using this advisory difference metric, constraint (10) requires that the two advisories in each pair of returned advisories utilize different airspace configurations during at least d time steps. This metric was developed based on discussions with area supervisors in which they identified differences in airspace configurations as operationally-significant. The area supervisors did not regard differences in operating position or workstation configurations corresponding to the same airspace configuration as significant.

C. Related Research

To clarify our contributions and to further motivate our approach for finding multiple advisories, we will describe some related research. Some of this literature review is also contained in our previous work, in which we motivate the M-ε-d-CSAs problem more exhaustively [12].

In the context of "planning" in the artificial intelligence domain, Nguyen et al. have investigated situations in which user preferences are unknown or when only a distribution over user objective function parameter values is provided [11]. Neither of these exactly describe the situation we investigate in this article, but this work helps motivate the M-ε-d-CSAs problem statement. In general, Nguyen et al. propose presenting the user with options and then allowing the user to

resolve uncertainty in the objective function by selecting a solution from the set of options. Similarly, implicit in the M-ε-d-CSAs problem statement is the assumption that the user will resolve the impact of unmodeled or imperfectly-modeled components of the area configuration problem by selecting an advisory option that performs well enough with respect to these components.

More specifically, when user preferences are completely unknown, Nguyen et al. suggest presenting the user with a *diverse* set of plans. If diversity is defined appropriately, diverse plans are less likely to be equally preferred by users, so a diverse set of plans increases the chances that one of the plans in the set will be acceptable to the user. In the area configuration context, user preferences are *not* completely unknown and various objective functions have been defined [2–9]. However, even the relatively complex objective function defined in [8] and [9] does not fully capture the relationship between configurations and safe and efficient traffic operations. Therefore, rather than striving only for diversity, the M-ε-d-CSAs problem statement requests a set of solutions that is both diverse and in which each plan performs well with respect to the objective function. The hope is that the diverse plans will achieve a variety of levels of performance with respect to unmodeled or imperfectly-modeled components of the problem so that at least one will perform well enough to be useful in operations.

The search for multiple low-cost and diverse paths has also been investigated in the operations research literature. Motivated by the “compromises" or “approximations" present in any mathematical model, Bellman and Kabala studied the problem of finding a given number of paths with the lowest combined cost [17]. This lowest-cost paths problem has received considerable attention in the literature [17–22], but simply searching for the lowest-cost paths without considering the diversity of the paths might lead to similar paths. For the reasons discussed earlier, we seek a set of near-optimal paths that are also diverse. For certain special definitions of diversity, the problem of finding a low-cost set of diverse paths has been studied [23–29]. For example, Suurballe proposes an algorithm that searches for low-cost paths that share no nodes [23]. The motivation provided in this body of research is often related to robustness—a path from the set should remain available even if some set of nodes or links fails. Unfortunately, subject-matter expert feedback suggests that none of the definitions of diversity used in this body of research correspond to the type of diver-

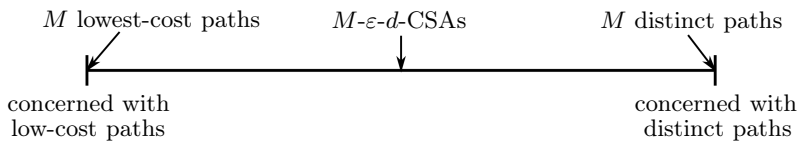


Fig. 4 One dimension of the space of problems involving finding M paths.

sity that matters for area configurations advisories (see sub-section III.B.1). This is not surprising because, although supervisors would likely benefit from advisories that are robust to uncertainty in the cost resulting from uncertainty in future air traffic, we are not concerned here with finding robust advisories.

Figure 4 is a notional visualization of one dimension of the space of problems involving finding a set of M paths. Problems concerned with finding low-cost paths, such as the lowest-cost paths problem, are at one end of the spectrum in this dimension. Problems concerned with finding distinct paths are at the other end of the spectrum. The distinct paths problem, which seeks a given number of paths that share no nodes without concern for path costs, is at this extreme of the spectrum. The M-ε-d-CSAs problem lies somewhere in the middle of the spectrum. It seeks low-cost paths (advisories), but at the same time requires that each returned path achieve a certain level of difference from the other returned paths. Efficient algorithms have been specified for some problems in this space, mostly at the ends of this spectrum, but we show in Appendix A that the M-ε-d-CSAs problem, located somewhere in the middle, is NP-hard.

One algorithm from the literature that proved useful in this article was proposed by Byers and Waterman [14]. This dynamic programming-based algorithm finds all paths that achieve a cost within some fraction of the minimum cost of a path. Once this set is known for a given excess cost fraction bound ε, we can solve the M-ε-d-CSAs problem by searching through the set for the M lowest-cost paths that meet the other problem constraints. Although computationally expensive, this approach can serve as a benchmark: it enables us to study how well more computationally-efficient heuristics perform on some small problem instances.

Although this article focuses on using time-expanded graphs to find area configuration schedule advisories, the approaches developed here may also be useful for other applications where multiple

low-cost paths are desired on time-expanded graphs. For example, the solution to a lowest-cost paths problem on a time-expanded graph has been used to generate multiple itineraries for the earliest arrival problem [30]. The approaches developed here might be able to find a set of *distinct* itineraries with early arrival times for some appropriate definition of distinctness.

IV. Algorithms

Three algorithms that solve or attempt to solve the M-ε-d-CSAs problem have been developed. One of these, described in sub-section IV.A, is a trivial extension to an algorithm proposed by Byers and Waterman [14]. It returns an optimal solution and serves as a benchmark. Sub-section IV.B describes a novel heuristic that is based on value iteration. A class of novel heuristics based on the A^* algorithm [13] is described in sub-section IV.C, culminating in the specification of the third algorithm. In addition to these three algorithms, we studied the solution to the M lowest-cost paths relaxation of M-ε-d-CSAs. This is described in sub-section IV.D. Finally, in sub-section IV.E, we summarize the computational complexity of these algorithms.

A. Value Iteration Fraction Optimal with Exhaustive Advisory Search

The Value Iteration Fraction Optimal with Exhaustive Advisory Search (VIFOEAS) algorithm is an extension of the algorithm proposed by Byers and Waterman [14] for finding all paths with costs that satisfy a bound on the excess cost fraction. VIFOEAS is specified in Algorithm 1. The first step is to use the well-known reverse value iteration algorithm to find optimal costs-to-go; it is specified in Appendix B for completeness. More concretely, it is denoted as ReverseVI($\mathcal{C}$) and it returns two mappings: (1) the minimum cost-to-go, $\bar{J}_k^\star(C_k)$, from each valid configuration and (2) the next configuration, $\bar{C}_{k+1}^\star(C_k)$, in a minimum-cost partial advisory starting from each configuration. Next, a recursive implementation of the algorithm proposed in [14] is utilized to find all the valid advisories that achieve the excess cost fraction bound constraint (9). We utilize a recursive function referred to as Recursive Value Iteration Fraction Optimal (RecursiveVIFO) to implement the algorithm proposed in [14] (see Appendix C). RecursiveVIFO returns the set $\mathcal{C}^\varepsilon \subseteq \mathcal{C}$ of advisories that are valid for the corresponding CSA problem instance and achieve costs within the fraction ε of the minimum cost. Next, by searching exhaustively through $\mathcal{C}^\varepsilon$ to find the cost-

minimizing M advisories that meet the other M-ε-d-CSAs constraints, the VIFOEAS algorithm solves the M-ε-d-CSAs problem.

Algorithm 1 Value Iteration Fraction Optimal with Exhaustive Advisory Search (VIFOEAS)

Require: $\mathcal{C}, T, M, \varepsilon, d$ {M-ε-d-CSAs problem instance specification}

$\{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}, \{\bar{C}_{k+1}^\star(C_k)\}_{k=0}^{K-1} \leftarrow \text{ReverseVI}(\mathcal{C}, T)$

$\mathcal{C}^\varepsilon \leftarrow \text{RecursiveVIFO}(\mathcal{C}, \{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}, C_0, (1+\varepsilon) \times \bar{J}_0^\star(C_0), 0)$

$J_{\min}^M \leftarrow \infty$

for each advisory set $\mathcal{C}^M \subseteq \mathcal{C}^\varepsilon$ such that $|\mathcal{C}^M| = M$ and $C^1 \in \mathcal{C}_{\text{CSA}}^\star(\mathcal{C}, T)$ **do**

  if $\min_{\{C^m, C^{m'} \in \mathcal{C}^M | m \neq m'\}} \Phi(C^m, C^{m'}) \geq d$ **and** $\sum_{C^m \in \mathcal{C}^M} g(C^m, T) \leq J_{\min}^M$ **then**

    $J_{\min}^M \leftarrow \sum_{C^m \in \mathcal{C}^M} g(C^m, T)$

    $\mathcal{C}_{\min}^M \leftarrow \mathcal{C}^M$

return $\mathcal{C}_{\min}^M$

Since M-ε-d-CSAs is NP-hard, it is not surprising that both finding the set of advisories that meet the cost constraint (9) and searching through these advisories for the M that solve the problem can be computationally demanding. Building the set of near-optimal advisories $\mathcal{C}^\varepsilon$ involves first performing backwards value iteration and then a depth-first search. If $n = \max_{k \in 1,\ldots,K} |\mathcal{C}_k|$ is the largest number of valid configurations at any configuration time step, then the time-expanded graph corresponding to a CSA problem instance has at most $nK+1$ nodes and n^2K+n edges. For a general graph, the computational complexity of value iteration is $\mathcal{O}((nK+1)(n^2K+n)) = \mathcal{O}(n^3K^2)$ [16]. However, the special time-expanded structure of the graph we are studying reduces the number of computations required. Value iteration in this case involves, for each of K time steps, performing a minimization over at most n values for at most n nodes, so its computational complexity is only $\mathcal{O}(n^2K)$ (see Appendix B). In the worst case, the depth-first search algorithm will visit each node once along each of the n edges into the node, and each time the node is visited, proceed to each of the n other nodes that can be reached from the node. This implies n^2 computations per node and there are nK nodes (other than C_0), leading to a computational complexity of $\mathcal{O}(n^3K)$ for the depth-first search and a total complexity of $\mathcal{O}(n^2K + n^3K) = \mathcal{O}(n^3K)$. Of course the computational effort required for the depth-first search varies considerably from problem instance to problem instance, depending largely on the degree of near-optimality desired. The computational

and memory efficiency of the depth-first search could be improved with a stack implementation, but we utilized this recursive implementation because it was simpler to implement. Once $\mathcal{C}^\varepsilon$ has been determined, searching through it requires the investigation of $\binom{|\mathcal{C}^\varepsilon|}{M}$ subsets of advisories. This search can usually be computed more efficiently than the brute-force search described in Algorithm 1 because the optimal advisory is typically unique. For example, the search is only linear in the size of $\mathcal{C}^\varepsilon$ when the optimal advisory is unique and $M = 2$. Although implemented, these simpler and more efficient variations of the search through $\mathcal{C}^\varepsilon$ are not precisely documented in this article.

B. Forward and Backward Value Iteration with Sequential Advisory Search

The Forward and Backward Value Iteration with Sequential Advisory Search (FBVISAS) algorithm is a heuristic that reduces the number of advisories to investigate by using minimum cost-to-go and minimum cost-so-far information. More precisely, it stores each node in a priority queue $\mathcal{J}$, ranked from smallest to largest sum of the minimum cost-so-far and the minimum cost-to-go. The nodes are then searched in the order specified by the priority queue until M configuration advisories that meet the constraints are found. The specification of FBVISAS is in Algorithm 2. This algorithm makes use of the reverse value iteration algorithm specified in Appendix B as well as the forward value iteration algorithm, which is referred to as ForwardVI. ForwardVI is completely analogous to ReverseVI except that it operates forward in time through the configuration time steps.

FBVISAS requires value iteration to be performed once in each direction, inducing a computational complexity of $\mathcal{O}(2n^2K)$. Then each of up to $nK + 1$ nodes must be investigated and inserted into the priority queue (with a computational complexity of up to $\mathcal{O}(\log(nK + 1))$ each time for a heap implementation). Finally, all $nK + 1$ nodes may need to be removed from the priority queue, which introduces a computational complexity of up to $\mathcal{O}(\log(nK + 1))$ for each node, again assuming a heap implementation. The total computational complexity is thus $\mathcal{O}(2n^2K + 2(nK + 1)\log(nK + 1)) = \mathcal{O}(n^2K + nK\log(nK + 1))$. If it is not possible to find M advisories that satisfy the constraints, the algorithm returns as many as it can find.

FBVISAS can be expected to perform relatively well on problem instances in which the reconfiguration cost is relatively important compared to the static cost. This could occur when β^R is

Algorithm 2 Forward and Backward Value Iteration with Sequential Advisory Search (FBVISAS)

Require: $\mathcal{C}, T, M, \varepsilon, d$ {M-ε-d-CSAs problem instance specification}

$\mathcal{C}^M \leftarrow \emptyset$

$\{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}, \{\bar{C}_{k+1}^\star(C_k)\}_{k=0}^{K-1} \leftarrow$ ReverseVI$(\mathcal{C}, T)$

$\{\underline{J}_k^\star(C_k)\}_{k=1}^{K}, \{\underline{C}_{k-1}^\star(C_k)\}_{k=1}^{K} \leftarrow$ ForwardVI$(\mathcal{C}, T)$

for $k = 1, \ldots, K$ **do**

  for $C_k \in \mathcal{C}_k$ **do**

    $J^\star(C_k) \leftarrow \underline{J}_k^\star(C_k) + \bar{J}_k^\star(C_k)$

    Insert C_k into priority queue $\mathcal{J}$ with key $J^\star(C_k)$

$C_k \leftarrow$ minimum-key configuration in $\mathcal{J}$ {Use to seed construction of minimum-cost advisory}

$J^\star \leftarrow$ last extracted key $J^\star(C_k)$

Use $\underline{C}_{k-1}^\star(C_k)$ and $\bar{C}_{k+1}^\star(C_k)$ iteratively to define minimum-cost advisory C^1

Add C^1 to $\mathcal{C}^M$

repeat

  $C_k \leftarrow$ minimum-key configuration in $\mathcal{J}$

  $J^\star \leftarrow$ last extracted key $J^\star(C_k)$

  Use $\underline{C}_{k-1}^\star(C_k)$ and $\bar{C}_{k+1}^\star(C_k)$ iteratively to define advisory $\tilde{C}$

  if $\Phi(C^m, \tilde{C}) \geq d$ for all $C^m \in \mathcal{C}^M$ **then**

    Add $\tilde{C}$ to $\mathcal{C}^M$

until $|\mathcal{C}^M| = M$ or $|\mathcal{J}| = 0$

return $\mathcal{C}^M$

relatively large and/or d is relatively small. We will make this statement more precise by defining two classes of problem instances. These classes are merely illustrative; neither of them are likely to be encountered when actually selecting area configuration advisories to present to supervisors. In one class, β^{R} is very large and we show that FBVISAS will find an optimal second advisory when one exists. In the other class, $\beta^{\mathrm{R}} = 0$ and we show that FBVISAS will never return a second advisory when $d > 1$. Appendix D contains proofs of these propositions.

C. Sequential Distinct A^* Algorithms

The Sequential Distinct A^* (SDA^*) algorithm is a novel heuristic that extends the well-known A^* algorithm for computing a lowest-cost path proposed by Hart et al. [13]. SDA^* runs an A^*-like

algorithm M times, using a different priority queue ranking function each time. The Sequential Distinct A^* with Shortcuts (SDA^*-SC) algorithm extends SDA^* by using information from the initial, time-reversed execution of the A^*-like algorithm to find low-cost and distinct advisories more quickly. Before describing the SDA^* and SDA^*-SC algorithms, we will specify the A^* and A^*-like algorithms that are utilized by these algorithms.

Although we introduced and evaluated the SDA^* algorithm in our previous work [12], the SDA^*-SC algorithm is a new contribution. Furthermore, in order to better motivate these algorithms, we will prove some properties of Forward Distinct A^* (FDA^*), an algorithm that is closely related to SDA^* and SDA^*-SC.

1. A^* Algorithm.

For completeness, Appendix E specifies the ForwardA^* algorithm, initially proposed by Hart et al. in [13], as applied to the CSA problem. The algorithm can be run to construct an advisory starting from C_0 and working forward in time (as specified in Appendix E), or vice versa. The algorithm makes use of $\hat{J}_k(C_k)$, an under-estimate of the minimum cost-to-go to a final configuration from a configuration C_k. A priority queue `open` and a set `closed` are also used. The configuration $\underline{C}^{\dagger}_{k-1}(C_k)$ is the previous configuration in a partial advisory under consideration. This previous configuration may or may not be part of a minimum-cost partial advisory. The $\underline{J}_k(C_k)$ values are upper bounds on the minimum cost-so-far required to get from C_0 to C_k. The cost $\underline{J}_k(C_k)$ can be achieved by constructing the partial advisory defined by the previous configurations returned by $\underline{C}^{\dagger}_{k-1}(C_k)$ back to C_0. Furthermore, using an under-estimate for $\hat{J}_k(C_k)$ ensures that once a configuration is moved to `closed`, $\underline{J}_k(C_k)$ for that configuration is actually the minimum cost $\underline{J}^{\star}_k(C_k)$ for a partial advisory from C_0 to C_k.

Since the cost for each time step is nonnegative ($g_k(C_{k-1}, T_{k-1}, C_k, T_k) \geq 0$) and $\hat{J}_k(C_k)$ is an underestimate of the minimum cost-to-go, this algorithm returns a lowest-cost advisory. We use $\hat{J}_k(C_k) \triangleq 0$, so A^* is a version of Dijkstra's algorithm that terminates as soon as a shortest path from C_0 to a configuration in $\mathcal{C}_K$ is found. Assuming a heap implementation of the priority queue structure, Dijkstra's algorithm has computational complexity $\mathcal{O}((n^2K + n)\log(nK + 1)) =$

$\mathcal{O}(n^2 K \log(nK+1))$ [16].

2. Forward Distinct A^* Algorithm.

The Forward Distinct A^* (FDA^*) algorithm is specified in Appendix F. FDA^* is not used by SDA^* or SDA^*-SC, but it is closely related to the Forward Distinct A^* with Shortcuts (FDA^*-SC) algorithm that they do utilize. FDA^* is related to the Lagrange dual problem for certain problem instances, so we will specify and discuss FDA^* to help explain and motivate FDA^*-SC. As specified, the FDA^* algorithm only finds a second advisory. It does so by finding an advisory that minimizes

$$g(C^2, T) + \lambda(d - \Phi(C^1, C^2)) \tag{13}$$

for some $\lambda \in \mathbb{R}_+$. This is the Lagrangian of the problem faced when finding the second advisory for some simple M-ε-d-CSAs instances. This property can be used to show that, under the right circumstances, the second advisory returned by FDA^* satisfies a necessary condition that must be met by any optimal second advisory. Some algorithms for constrained shortest path algorithms similarly leverage Lagrange duality [31, 32]. Appendix G provides proofs and further discussion of these properties of FDA^*.

3. Forward Distinct A^* with Shortcuts Algorithm.

The FDA^*-SC algorithm, which is specified in Appendix F, extends the FDA^* algorithm in four main ways. The first is that it explicitly describes how to handle problem instances for which $M > 2$. Second, it confirms that the partial advisory ending at a configuration for a given time step has a chance of meeting the cost and difference constraints before adding the configuration to the queue of configurations and corresponding partial advisories under consideration. Third, it normalizes the terms in the cost function to increase the likelihood that a single value for λ will perform well across problem instances. In particular, while FDA^* finds a second advisory that minimizes the objective (13), FDA^*-SC attempts to minimize

$$\frac{g(C^m, T)}{J^\star} + \lambda \frac{1}{m-1} \sum_{m'=1}^{m-1} \frac{d - \Phi(C^{m'}, C^m)}{\Phi_{\max} - d + 1}$$

when searching for the m^{th} advisory. Here $\Phi_{\max}$ is the maximum possible advisory difference metric value. Showing that FDA^*-SC attempts to minimize this quantity requires some algebraic

manipulations similar to those used in the proof of Lemma 3 in Appendix G. Normalizing by $J^\star$, $(m-1)$, and $(\Phi_{\max}-d+1)$ is designed to increase the likelihood that the magnitude of the first term and the magnitude of the term multiplied by λ will not vary much across problem instances and as we search for subsequent advisories. The hope is that this will ensure that a single value of λ may be sufficient in all of these contexts and we can avoid re-tuning λ for each problem instance or for searches for each subsequent advisory.

Finally, the fourth change is that the FDA^*-SC algorithm can take shortcuts in an attempt to reduce computation times. The FDA^*-SC algorithm makes use of information that could be provided by an initial ReverseA^* run that could also find the first returned advisory. In particular, this initial run would yield partial advisory and corresponding cost information from certain configurations at certain time steps to the final time step. This information is in $\{\bar{J}_k(C_k)\}_{k=0}^{K-1}$ and $\{\bar{C}_{k+1}^{\dagger}(C_k)\}_{k=0}^{K-1}$, which are not necessarily defined for all C_k. When investigating a partial advisory ending at some C_k, FDA^*-SC checks to see if, by combining this partial advisory from the initial time step with the partial advisory that continues on from C_k and was explored by the initial ReverseA^* run, we arrive at a complete *shortcut* advisory that (1) achieves an acceptable cost value (as determined by the algorithm parameter $\varepsilon' \in [0, \varepsilon]$) and (2) also achieves the difference constraint. If this is the case, then the algorithm returns a shortcut advisory the next time it queries the priority queue.

4. Sequential Distinct A^* Algorithm with Shortcuts.

The SDA^*-SC algorithm is specified in Algorithm 3. This algorithm involves using ReverseA^* to find the first advisory and then FDA^*-SC to find any subsequent advisories. Since SDA^*-SC is essentially M iterations of variations of A^*, its computational complexity is $\mathcal{O}(Mn^2K\log(nK+1))$. If $\varepsilon' = 0$, then the algorithm never takes a shortcut and we refer to it as just SDA^*.

D. Lowest-Cost Paths

If we relax the M-ε-d-CSAs problem by disregarding constraints (9) and (10), we are left with an M lowest-cost paths problem. Many efficient algorithms that solve this problem have been developed [17–22]. For example, the computational complexity of the algorithm proposed by Eppstein is $\mathcal{O}(n^2K + nK\log(nK+1) + M)$, which is essentially identical to the complexity of the

Algorithm 3 Sequential Distinct A^* with Shortcuts (SDA^*-SC)

Require: $\mathcal{C}, T, M, \varepsilon, d$ {M-ε-d-CSAs problem instance specification}

Require: λ, ε' {Algorithm parameters}

$C^1, \{\bar{J}_k(C_k)\}_{k=0}^{K-1}, \{\bar{C}_{k+1}^{\dagger}(C_k)\}_{k=0}^{K-1} \leftarrow \text{Reverse}A^*(\mathcal{C}, T)$

Add C^1 to $\mathcal{C}^M$

$J^\star \leftarrow g(C^1, T)$

for $m = 2, \ldots, M$ **do**

$\mathcal{C}^M \leftarrow \text{FD}A^*\text{-SC}(\mathcal{C}, T, J^\star, \varepsilon, d, \mathcal{C}^M, \lambda, \{\bar{J}_k(C_k)\}_{k=0}^{K-1}, \{\bar{C}_{k+1}^{\dagger}(C_k)\}_{k=0}^{K-1}, \varepsilon')$

if $|\mathcal{C}^M| < m$ **then**

Skip remainder of **for** loop {Do not attempt to construct the remaining $M - m$ advisories}

return $\mathcal{C}^M$

Table 1 Computational complexity of algorithms

Algorithm	Complexity	Reference
VIFOEAS	$\mathcal{O}(n^3K)$[a]	sub-section IV.A
FBVISAS	$\mathcal{O}(n^2K + nK\log(nK+1))$	sub-section IV.B
SDA^*-SC	$\mathcal{O}(Mn^2K\log(nK+1))$	sub-section IV.C
Eppstein lowest-cost paths	$\mathcal{O}(n^2K + nK\log(nK+1) + M)$	[20]
Suurballe lowest-cost node-disjoint paths	$\mathcal{O}(Mn^2K\log(nK+1))$[b]	[23]

[a] This does not include the complexity of searching through $\binom{|\mathcal{C}^\varepsilon|}{M}$ advisory subsets.

[b] This assumes that Dijkstra's algorithm is used as a subroutine for finding shortest paths.

FBVISAS heuristic even though the former algorithm is optimal (albeit for the lowest-cost paths relaxation of the problem of interest). Therefore, we investigated the solution to the relaxed M lowest-cost paths problem version of some M-ε-d-CSAs problem instances to see if the returned advisories happened to meet the constraints (9) and (10). If this occurred frequently enough, then one of these existing algorithms for the lowest-cost paths problem could be used for the M-ε-d-CSAs problem. We implemented an algorithm for the M lowest-cost paths problem and refer to it as the LCP algorithm.

E. Computational Complexity Comparison

The computational complexities of various algorithms proposed and discussed in this article are documented in Table 1. The complexity is reported when n is the largest number of valid configurations in configuration time steps $k = 1, 2, \ldots, K$, which means that the graph under consideration has at most $nK + 1$ nodes and $n^2K + n$ edges when we also account for the initial configuration C_0. The complexity of value iteration for the graph under consideration is lower than for a general graph, and this is reflected in the complexity results for VIFOEAS and FBVISAS. The complexities of the other algorithms do not leverage the special structure of the graph under consideration (i.e., they are based only on the number of nodes and edges in the graph, not on the time-expanded structure). This, along with the worst-case assumption of that no shortcuts are found, may explain why the complexity of SDA^*-SC is larger than that of FBVISAS even though we show in sub-section V.B that SDA^*-SC always executed in less time than FBVISAS over thousands of problem instances. As expected, the complexity of VIFOEAS is larger than that of the other algorithms, and it does not even include the cost of searching through up to $\binom{|\mathcal{C}^\varepsilon|}{M}$ advisory subsets. The complexities of two representative algorithms for finding M paths are provided for reference [20, 23] (see sub-section III.C). Each is optimal for a particular problem that involves finding M paths. The lowest-cost paths algorithm solves a relaxation of M-ε-d-CSAs (see sub-section IV.D). The lowest-cost node-disjoint paths algorithm solves a problem similar to the M-ε-d-CSAs problem but with a different ϕ function, a particular value of d, and in which constraint (8) is not enforced (i.e., the first advisory is not required to be optimal for a single-path problem). The complexity of the lowest-cost paths algorithm is essentially identical to that of FBVISAS (particularly for $M = 2$ or 3), and the complexity of the lowest-cost node-disjoint paths algorithm is identical to that of SDA^*-SC. The complexities of the FBVISAS and SDA^*-SC heuristics are therefore comparable to those of algorithms that are optimal for other problems that are related to the M-ε-d-CSAs problem.

V. Performance Analysis

The performance of the algorithms was first analyzed by using them to solve some small problem instances for which optimal solutions could be computed by the benchmark VIFOEAS algorithm in

a reasonable amount of time; this analysis is documented in sub-section V.A. Then, as described in sub-section V.B, we analyzed the performance of the two novel heuristics by using them to solve thousands of realistic problem instances.

Before presenting the results of the performance analysis, we will provide some notes about our implementation of the algorithms. The algorithms were coded in Java and executed on a MacPro workstation with a Quad-Core Intel Xeon 2.8 GHz processor and 4 GB of memory. Before a problem instance is specified or an algorithm is executed, the set of all possible configurations that could be used in a time step is pre-computed and stored for use by all instances. Specifying a problem instance involves tasks like fetching traffic data, parsing parameter files, and translating constraints into a set of configurations that could be used at some time step by an advisory. These instance-specification tasks are performed using the same code regardless of which algorithm is deployed, so the time required for these tasks does not vary between algorithms. When an algorithm first calls the single-time step cost function $g_k(C_{k-1}, T_{k-1}, C_k, T_k)$ for a set of inputs, the cost value is computed and then cached. If the algorithm later requires the single-time step cost for the same set of inputs, the cached cost value is returned, which reduces the computational burden of the algorithms.

A. Investigation of the Performance of Algorithms on Small Problem Instances

In this sub-section, we investigate the LCP algorithm and the FBVISAS, SDA^*, and SDA^*-SC heuristics by comparing their output to that of the VIFOEAS algorithm, which is guaranteed to return an optimal solution, when they are asked to solve some small problem instances. The small problem instances utilized here are based on ZOB area 4 (see Section II and Fig. 1(a)). The instances come from nine two-hour time periods (6 am–8 am, 8 am–10 am, . . ., 10 pm–midnight local time) on two dates (Thursday 01 December 2011 and Tuesday 06 December 2011), so there are 18 instances. The configuration time step was five minutes ($\Delta = 5$) and the traffic time step was one minute ($\delta = 1$). The algorithm was restricted to select from 16 airspace configurations that ZOB staff have identified as feasible. Default workstation assignments were used for all open sectors. There were 173 area configurations (i.e., corresponding airspace, operating position, and workstation configurations)

that could be generated from these 16 airspace configurations. However, the advisories were required to use the same *number* of open sectors as were used in each configuration time step during actual operations on those days. Therefore, there were only between 4 and 80 valid configurations available at each time step instead of 173, which reduced the number of advisories in $\mathcal{C}$ considerably. In order to further reduce the size of the space of feasible solutions, only two advisories were requested ($M = 2$) and the second advisory was required to achieve a cost value relatively close to the minimum cost ($\varepsilon = 0.2$). Based on parameter-tuning efforts documented in [8] and [9], the cost function parameter β^{R} was set to 2. Finally, based on discussions with subject-matter experts, d was set equal to 6. This implies that different airspace configurations must be used for at least 30 of the 120 minutes in the two returned advisories. The SDA^* parameters for these problem instances were set to $\lambda = 0.11875$ and $\varepsilon' = 0$. The SDA^*-SC parameters were set to $\lambda = 0.11875$ and $\varepsilon' = 0.2$.

All of the algorithms found the same minimum-cost first advisory for all of the 18 instances. The VIFOEAS algorithm found two advisories for 12 of the 18 instances. The LCP algorithm found two feasible advisories for only 1 of the 18 instances, indicating that relaxing our problem by ignoring the constraints (9) and (10) so that we can solve an M lowest-cost paths problem frequently does not lead to M feasible advisories for the M-ε-d-CSAs problem, even when M such advisories exist. The FBVISAS, SDA^*, and SDA^*-SC heuristics each found two advisories for 10, 12, and 12 instances, respectively. These results suggest that the novel heuristics typically find a feasible second advisory when one exists.

Since the first advisories returned by the algorithms all achieve the same cost, the suboptimality of the advisories provided by each algorithm was investigated by focusing on the costs of the second advisories. For problem instances with a feasible second advisory, Fig. 5 shows the excess cost fraction achieved by the second advisory returned by each of the algorithms. An empty marker located at fraction 0.30 indicates that an algorithm failed to return a second advisory for a problem instance. By comparing the second advisory returned by each algorithm to the second advisory returned by the VIFOEAS algorithm, which is guaranteed to achieve the lowest cost of all the second advisories that satisfy the M-ε-d-CSAs constraints, we see that the FBVISAS, SDA^*, and SDA^*-SC algorithms each found a lowest-cost second advisory in 9, 9, and 5 instances, respectively.

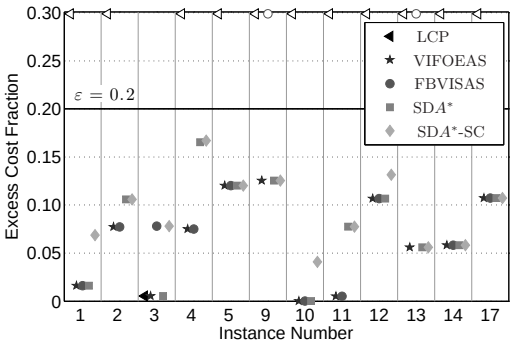


Fig. 5 Excess cost fraction achieved by second advisories.

These results suggest that the FBVISAS and SDA^* heuristics are better at finding a lowest-cost second advisory than the SDA^*-SC heuristic. This result is to be expected because of the shortcuts that the SDA^*-SC heuristic takes in an attempt to reduce computation times.

When generating these 18 instances 5 times each (once per algorithm), the mean and standard deviation of time required to load the traffic data were 856 ms and 104 ms, respectively. The mean and standard deviation of the time required for other instance-specification tasks were 35 ms and 4 ms, respectively. However, these problem instances are too small to allow for a meaningful analysis of the computation time required by the algorithms, with one exception. These instances reveal that the optimal VIFOEAS algorithm is far more computationally intensive than other algorithms. It required hours to find a solution for some of these problem instances, while other algorithms required less than a second. Given that we desire to solve larger problem instances than are studied in this sub-section in less than a minute on a tablet computer, these results indicate that the computational performance of VIFOEAS is unacceptable. Even if a more efficient stack implementation were used (see sub-section IV.A), it is unlikely that VIFOEAS would achieve acceptable computation times.

B. Investigation of the Performance of Algorithms Using a Year of Data

To better understand their behavior on realistic problem instances, the FBVISAS and SDA^*-SC heuristics were used to solve many more problem instances. The instances involve ZOB area 4 for 231 non-weekend and non-holiday days selected from 20 October 2011 to 19 October 2012. Weekends and holidays were excluded because they might involve low-volume or atypical traffic

Table 2 Fraction of problem instances with one, two, or three advisories returned

Algorithm	One Advisory	Two Advisories	Three Advisories
SDA^*-SC	0.21	0.20	0.59
FBVISAS	0.15	0.20	0.66

patterns, leading to configuration selections that are correspondingly atypical. On each day, 18 problem instances were solved for the time periods from 6:00 am–8:00 am, 7:00 am–9:00 am, ..., 11:00 pm–1:00 am local time. These overlapping problem instances were selected to approximate how area configuration advisories might be used in practice. A total of 4158 instances were solved by each of the two heuristics.

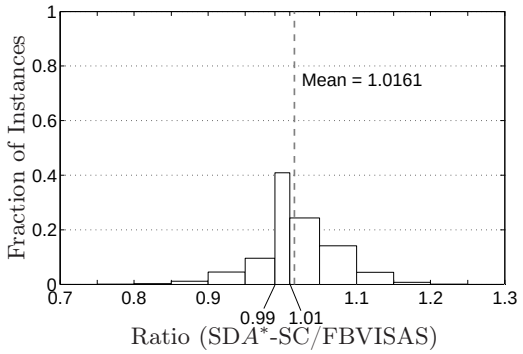
For these instances, the following parameter values were selected: $\beta^{\mathrm{R}} = 2$, $M = 3$, $d = 6$, and $\varepsilon = 0.5$. The configuration time step was five minutes ($\Delta = 5$) and the traffic time step was one minute ($\delta = 1$). Furthermore, SDA^*-SC parameters were set to $\lambda = 0.11875$ and $\varepsilon' = 0.25$. The same set of 173 configurations were permitted in these instances as in those described in subsection V.A, but in this case the only additional constraint was that the initial airspace configuration C_0^{A} be identical to the airspace configuration that historical records indicate was in use at the start of the problem instance.

Table 2 shows the fraction of problem instances for which each heuristic returned one, two, or three advisories. The FBVISAS heuristic is more likely to return more advisories. It returns three advisories in 7% more instances and one advisory in 6% fewer instances than the SDA^*-SC heuristic.

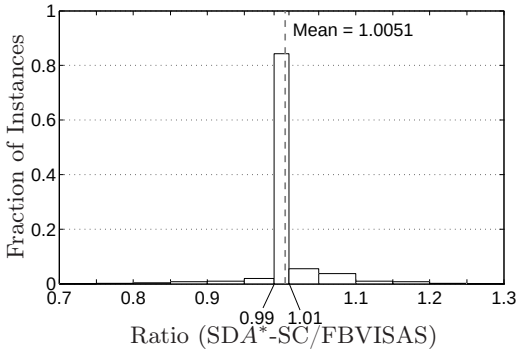
Both algorithms found exactly two advisories in 508 of the 4158 problem instances and both found three advisories in 2380 of these instances. The relative value of the costs of the returned advisories were investigated for these two classes of instances. When $M = 3$, the M-ε-d-CSAs problem objective (5) is to minimize $g(C^1, T) + g(C^2, T) + g(C^3, T)$, but constraint (8) requires that $g(C^1, T)$ be the same for any feasible solution. Therefore, more insight into the relative quality of solutions is gained by investigating only the costs of the second and third returned advisories. For instances where both heuristics returned three advisories, we studied the $g(C^2, T) + g(C^3, T)$ ratio: the sum of the costs of the second and third advisories returned by SDA^*-SC divided by

the sum of the costs of the second and third advisories returned by FBVISAS. For instances where both heuristics returned exactly two advisories, we study the $g(C^2, T)$ ratio: the cost of the second advisory returned by SDA^*-SC divided by the cost of the second advisory returned by FBVISAS.

Figures 6(a) and 6(b) show the distributions of these ratios. The ratio is between 0.99 and 1.01 in more than 40% of the instances where both heuristics returned three advisories and in more than 80% of the instances where both heuristics returned exactly two advisories. On average, the ratio is only slightly above 1 in each case. While there are a few instances where SDA^*-SC returned advisories that are 20% or more costlier than the advisories found by FBVISAS, these instances were relatively rare and the ratio never exceeded 1.3. SDA^*-SC did sometimes achieve lower costs for these second and third advisories, but it was more common for FBVISAS to find lower-cost advisories. Overall, FBVISAS tended to find slightly lower-cost advisories than SDA^*-SC.



(a) Sum of costs of second and third advisories $(g(C^2, T) + g(C^3, T))$.



(b) Cost of second advisory $(g(C^2, T))$.

Fig. 6 Distribution of the ratio of costs.

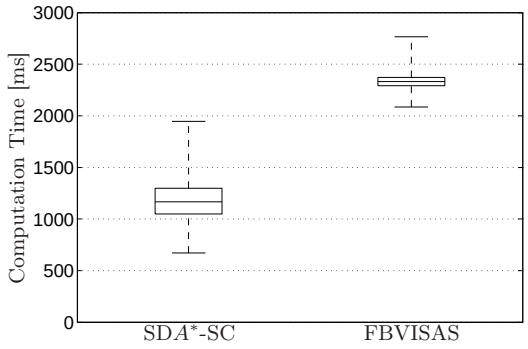


Fig. 7 Distributions of computation time required by the SDA^*-SC and FBVISAS heuristics.

When generating these 4158 instances 2 times each (once per heuristic), the mean and standard deviation of time required to load the traffic data were 824 ms and 95 ms, respectively. The mean and standard deviation of the time required for other instance-specification tasks were 38 ms and 3 ms, respectively. Figure 7 is a box plot showing the spread in the computation times of the two heuristics. SDA^*-SC always has lower computation times than FBVISAS. The average SDA^*-SC computation time (1283 ms) is just over half of the average FBVISAS computation time (2332 ms).

VI. Conclusions

It is difficult to model the relationship between area configurations and safe and efficient traffic operations, so we propose presenting area supervisors with a set of near-optimal and meaningfully-different configuration schedule advisories. To this end, we studied the M ε-optimal d-distinct Configuration Schedule Advisories problem, which is a significant extension of the Configuration Schedule Advisories problem for finding a single advisory [8, 9]. This problem is equivalent to finding an optimal path as well as other near-optimal and distinct paths in a time-expanded graph. It is related to the well-known lowest-cost paths problem but we showed that the constraint requiring distinct paths makes it NP-hard.

We proposed three algorithms for this problem and also investigated a fourth that solves the M lowest-cost paths problem, which is a relaxation of this problem. The Value Iteration Fraction Optimal with Exhaustive Advisory Search algorithm trivially extends the dynamic programming-

based algorithm proposed by Byers and Waterman [14]. Although it is computationally expensive, it finds an optimal solution: the lowest-total-cost set of advisories that meet the problem constraints. We proposed the novel Forward and Backward Value Iteration with Sequential Advisory Search heuristic, which is based on value iteration. Primarily by changing the value of a cost function parameter, we defined a class of problem instances for which the heuristic finds an optimal solution and another class of instances for which it will fail to return a second advisory even when a feasible second advisory exists. We also proposed the novel Sequential Distinct A^* with Shortcuts heuristic, which is a modification of the A^* algorithm and an extension of the Sequential Distince A^* heuristic we proposed in [12]. It can be motivated by studying the Lagrangian of certain simple problem instances and it is designed to achieve relatively low computation times. The computational complexities of these two novel heuristics are comparable to those of representative optimal algorithms for related problems that involve finding a set of paths.

We first evaluated the algorithms on small problem instances by comparing their solutions to the solution returned by the benchmark optimal algorithm. These problem instances revealed the inadequacy of the lowest-cost paths algorithm (it rarely returned feasible second advisories) and the benchmark optimal algorithm (it required excessive computation times). The two heuristics, on the other hand, typically returned feasible solutions when a feasible solution existed and found optimal solutions for half of the instances. The value iteration- and A^*-based heuristics were also used to solve thousands of realistic problem instances based on Cleveland Air Route Traffic Control Center Area of Specialization 4. For these instances, the average computation time for the A^*-based heuristic (1283 ms) was just over half of the average computation time of the value iteration-based heuristic (2332 ms). However, the value iteration-based heuristic found slightly lower-cost advisories (less than 2% lower on average). The value iteration-based heuristic also returned the problem-requested number of advisories (three) for 66% of the instances, while the A^*-based heuristic only returned three advisories for 59% of the instances. These results indicate that, when compared with the A^*-based heuristic, the value iteration-based heuristic offers higher-quality solutions at the expense of longer computation times.

Appendix A: The NP-Hardness of M-ε-d-CSAs

The M-ε-d-CSAs problem is NP-hard. This will be shown by demonstrating that the Independent Set (IS) problem, which is NP-complete and even difficult to approximate [15], is polynomial-time reducible to M-ε-d-CSAs. This is sufficient to show that the M-ε-d-CSAs problem is NP-hard [16].

First, a review of the IS problem will be provided. Given a graph $G = (V, E)$, a set of nodes are *independent* if none of the nodes are joined by an edge in G. The decision version of the IS problem is to report whether it is possible to find a set of nodes $U \subset V$ of size $|U| = M$ such that all of the M nodes are independent on G.

Now IS will be shown to be polynomial-time reducible to M-ε-d-CSAs. This will be done by demonstrating, for an arbitrary instance I_{IS} of the IS problem, how to construct, in polynomial time, an instance $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ of M-ε-d-CSAs such that

1. if I_{IS} is a "yes" instance of IS (that is, an independent set of the required size can be found), then there exists a feasible solution for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$, and

2. if there exists a feasible solution for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$, then I_{IS} is a "yes" instance of IS.

Given an arbitrary IS instance I_{IS} on a graph $G(V, E)$, the corresponding instance $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ is constructed in polynomial time as follows. The number of nodes M required in I_{IS} maps to the number of solutions M required in $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$. The cost function in $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ is defined to be equal to zero so that $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ becomes a feasibility problem ($g_k(\cdot,\cdot,\cdot,\cdot) \triangleq 0$). Then, a time-expanded graph is constructed for use in the $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ problem instance. For this graph, $K = 1$. This graph consists of a single starting node (a required starting configuration C_0 that makes up $\mathcal{C}_0$). For $k = 1$, any of the nodes V represent valid configurations ($\mathcal{C}_1 \triangleq V$). This means that any of the required M valid configuration schedule advisories will select a single node from V for C_1^m. The configuration difference metric ϕ is defined as follows:

$$\phi(C_k, C_k') = \begin{cases} 1 & \text{if } C_k \text{ and } C_k' \text{ do } \textit{not} \text{ share an edge in } G(V, E) \\ 0 & \text{else.} \end{cases}$$

The configuration difference metric is defined such that two configuration schedule advisories achieve

an advisory difference of 1 if and only if the configuration used by each advisory at $k = 1$ (which correspond to nodes from V) do not share an edge in the I_{IS} graph $G(V, E)$. Finally, the minimum required difference d for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ will be defined as $d \triangleq 1$.

Next, the first condition required of I_{IS} and $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ will be verified. Suppose that the arbitrary I_{IS} is a "yes" instance of IS: there exists a set of M nodes $U \subset V$ such that each node in U is independent from all of the other nodes in U. The set U can be converted into M configuration schedule advisories that make up a feasible solution for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$. The m^{th} configuration schedule advisory is defined with the required C_0 and with C_1^m equal to the m^{th} node from U. Since U is an independent set, each node in U shares no edge in E with any other node in U. This means that, due to the definition of the configuration difference metric for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$, the M configuration schedule advisories defined as described will meet the advisory difference constraint (10). These M configuration schedule advisories are feasible for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$, and the condition has been verified.

Finally, the second condition required of I_{IS} and $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$ will be verified. Suppose that there exists a feasible solution for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$. Let this feasible solution be $\{C^1, \ldots, C^M\}$, where each C^m is a configuration schedule advisory: $C^m = \{C_0^m, C_1^m\}$ for $m = 1, \ldots, M$. A "yes" instance of I_{IS} can be constructed as follows. Let $U = \{C_1^m\}_{m=1}^M \subset V$ be a set of size M that we will use to demonstrate that there exists an independent set of size M for I_{IS}. Since $\{C^1, \ldots, C^M\}$ make up a feasible solution for $I_{M\text{-}\varepsilon\text{-}d\text{-CSAs}}$, we know that they satisfy the advisory difference constraint. Due to how the configuration difference metric is defined, this means that each of the C_1^m do not share an edge in E with any other $C_1^{m'}$. This means that $U = \{C_1^m\}_{m=1}^M$ is an independent set of size M on $G(V, E)$, verifying that I_{IS} is indeed a "yes" instance of IS.

If we attempt to further show that M-ε-d-CSAs is NP-complete, we must also show that it is in NP, the set of problems for which possible solutions can be efficiently certified. We have not been able to show this, so we only know that the problem is NP-hard.

Appendix B: Reverse Value Iteration Algorithm

The Reverse Value Iteration algorithm (ReverseVI) is described in Algorithm 4.

Algorithm 4 ReverseVI($\mathcal{C}, T$)

Require: $\mathcal{C} = \{\mathcal{C}_k\}_{k=0}^{K}$ {Valid configuration schedule advisories}

Require: T {Traffic situation data}

for $C_K \in \mathcal{C}_K$ **do**

$\bar{J}_K^\star(C_K) \leftarrow 0$

for $k = K-1, \ldots, 0$ **do**

for $C_k \in \mathcal{C}_k$ **do**

$\bar{J}_k^\star(C_k) \leftarrow \min_{C_{k+1} \in \mathcal{C}_{k+1}} \left[g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1}) + \bar{J}_{k+1}^\star(C_{k+1}) \right]$

$\bar{C}_{k+1}^\star(C_k) \in \operatorname{argmin}_{C_{k+1} \in \mathcal{C}_{k+1}} \left[g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1}) + \bar{J}_{k+1}^\star(C_{k+1}) \right]$

return $\{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}, \{\bar{C}_{k+1}^\star(C_k)\}_{k=0}^{K-1}$

Appendix C: Recursive Value Iteration Fraction Optimal Algorithm

The Recursive Value Iteration Fraction Optimal algorithm (RecursiveVIFO) is described in Algorithm 5. It is a recursive implementation of the algorithm proposed by Byers and Waterman [14].

Algorithm 5 RecursiveVIFO($\mathcal{C}, \{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}, C_k, J_{\max}, \underline{J}$) Function

Require: $\mathcal{C} = \{\mathcal{C}_k\}_{k=0}^{K}$ {Valid configuration schedule advisories}

Require: $\{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}$ {Minimum cost-to-go values}

Require: $C_k \in \mathcal{C}_k$ {Last configuration in partial advisory under consideration}

Require: $J_{\max} \in \mathbb{R}_+$ {Upper bound on advisory cost}

Require: $\underline{J} \in \mathbb{R}_+$ {Cost incurred so far by the partial advisory under consideration}

$\mathcal{C}^\varepsilon \leftarrow \emptyset$

for $C_{k+1} \in \mathcal{C}_{k+1}$ **do**

if $\left[\underline{J} + g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1}) + \bar{J}_{k+1}^\star(C_{k+1}) \right] \leq J_{\max}$ **then**

if $k+1 = K$ **then**

Add partial advisory $\{C_k, C_K\}$ to $\mathcal{C}^\varepsilon$

else

$\mathcal{C}^{\varepsilon, C_{k+1}} \leftarrow$ RecursiveVIFO$(\mathcal{C}, \{\bar{J}_k^\star(C_k)\}_{k=0}^{K-1}, C_{k+1}, J_{\max}, \underline{J} + g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1}))$

for each partial advisory $\bar{C} \in \mathcal{C}^{\varepsilon, C_{k+1}}$ **do**

$\bar{C}' \leftarrow \{C_k, \bar{C}\}$ {Prepend C_k to the partial advisory $\bar{C}$}

Add $\bar{C}'$ to $\mathcal{C}^\varepsilon$

return $\mathcal{C}^\varepsilon$

Appendix D: Proofs of Properties of the Behavior the FBVISAS Heuristic on Two Classes of Problem Instances

Although more restrictive than is necessary, we will define and utilize *simple* problem instances.

Definition 1 (simple M-ε-d-CSAs problem instance) *A simple M-ε-d-CSAs problem instance is one in which*

- $M = 2$,
- $\varepsilon = \infty$,
- *the configuration constraints are such that* $\mathcal{C}_1 = \mathcal{C}_2 = \cdots = \mathcal{C}_K$ *and that* $C_0 \in \mathcal{C}_1$,
- *and there exists a unique optimal first advisory* ($|\mathcal{C}^{\star}_{CSA}(\mathcal{C}, T)| = 1$).

1. Simple Reconfiguration Cost-Dominated Problem Instances

Definition 2 (reconfiguration cost-dominated M-ε-d-CSAs problem instance) *Reconfiguration cost-dominated M-ε-d-CSAs problem instances are those in which β^R is large enough that*

$$\min_{k\in\{1,\ldots,K\}} \min_{\{C,C'\in\mathcal{C}_k | C\neq C'\}} \beta^R g^R_k(C, T_{k-1}, C', T_k) > \sum_{k=1}^{K} \max_{C,C'\in\mathcal{C}_k} [g^S_k(C, T_k) - g^S_k(C', T_k)].$$

The left hand side of this inequality is the lowest cost of changing configurations during the entire time horizon. The right hand side is the sum over all the configuration time steps of the largest difference between static costs at each time step. An important characteristic of simple reconfiguration cost-dominated instances is that advisories never achieve lower total costs by changing configurations—it is always better to remain in the current configuration.

Lemma 1 *The single optimal first advisory for simple reconfiguration cost-dominated problem instances uses the initial configuration for all time steps $k \in \{0, 1, \ldots, K\}$.*

Proof of Lemma 1 The definition of simple instances ensures that such an advisory meets the configuration constraints. The definition of reconfiguration cost-dominated instances ensures that lower advisory costs can always be achieved by staying in the same configuration rather than by

changing configurations. Therefore, using the initial configuration for the entire time horizon is the unique minimum-cost advisory. ∎

Proposition 1 *If one exists, FBVISAS finds an optimal second advisory for simple reconfiguration cost-dominated problem instances.*

Proof of Proposition 1 By Lemma 1, we know that the unique optimal first advisory uses the initial configuration $C_0 = C$ for the entire problem instance time horizon. The M-ε-d-CSAs problem statement requires that an optimal second advisory uses a different airspace configuration for at least d time steps. To meet this constraint, an optimal second advisory must reconfigure at some time step k to a new configuration C' such that $C^{\mathrm{A}} \neq C'^{,\mathrm{A}}$. We will investigate how FBVISAS handles this configuration C' at time step k to show by contradiction that it must return an optimal second advisory for these instances.

Before we investigate the operation of FBVISAS on these instances, we will establish some properties of an optimal second advisory. In particular, we will study two parts of an optimal second advisory: the partial optimal second advisory from C' in time step k to a final configuration in time step K and the partial optimal second advisory from C in time step 0 to C' in time step k.

- *Partial optimal second advisory from C' at time step k*: Consider the partial optimal second advisory from C' at time step k to the end of the time horizon at time step K. For the CSA problem instance corresponding to the M-ε-d-CSAs problem instance, the unique cost-minimizing partial advisory from C' at time step k until the final time step K will remain at C' because this is a simple reconfiguration cost-dominated instance. We will show that an M-ε-d-CSAs-optimal second advisory also remains at C' for the remainder of the time period. The change in configuration at time step k in this optimal second advisory must have occurred early enough to fulfill the difference constraint in the time steps from k to K. Since $C^{\mathrm{A}} \neq C'^{,\mathrm{A}}$, a partial advisory that remains at C' will therefore also achieve the difference constraint. Such a partial advisory also minimizes the M-ε-d-CSAs objective, so it is the portion of the optimal second advisory after time step k. Overall, we note that an M-ε-d-CSAs-optimal second advisory uses the unique CSA-minimal cost-to-go partial advisory from

C' at time step k, and this partial advisory achieves the minimum required difference from the optimal first advisory.

- *Partial optimal second advisory to C' at time step k*: Now consider the part of this optimal second advisory from C at time step 0 to C' at time step k. For the CSA problem corresponding to the M-ε-d-CSAs problem instance, the partial advisory that stays at C from time step 0 until time step $k-1$ and then changes to C' at time step k must be cost-minimizing among all partial advisories starting in C and ending in C' at time step k. Otherwise, a lower-cost second advisory for the M-ε-d-CSAs problem instance could be constructed by using this hypothetical other CSA cost-minimal partial advisory. Such a lower-cost second advisory would still meet the difference constraint (10) because that constraint is met by the portion of the advisory from time steps k to K. Overall, we note that the M-ε-d-CSAs-optimal second advisory achieves the CSA-minimal cost-so-far for partial advisories starting in C at time step 0 and ending in C' at time step k.

Now we leverage these two properties to show that FBVISAS will return an optimal second advisory. Arguing by contradiction, we assume that FBVISAS returned a second advisory with a larger total advisory cost than is achieved by an M-ε-d-CSAs-optimal second advisory. We have already shown that an M-ε-d-CSAs-optimal second advisory achieves the CSA-minimal cost-so-far from C at time step 0 to C' at time step k and the CSA-minimal cost-to-go from C' at time step k to the end of the time horizon. We have assumed that the cost of this optimal second advisory, which is the sum of the cost-so-far and cost-to-go, is lower than the cost of the second advisory returned by FBVISAS. FBVISAS investigates configurations at time steps starting with those that have the lowest sum of the cost-so-far and cost-to-go, so C' at time step k would have been investigated by FBVISAS *before* it found and returned the assumed higher-cost second advisory. However, if FBVISAS investigated C' at time step k, it would have discovered that the advisory constructed from these two partial advisories met the problem constraints because the partial advisory from C' at time step k meets the difference constraint on its own (as was shown earlier). Therefore, it would have returned this M-ε-d-CSAs-optimal second advisory, a result that contradicts our assumption that FBVISAS returned a second advisory with a larger total advisory cost than is achieved by an

M-ε-d-CSAs-optimal second advisory. Therefore, when one exists, FBVISAS returns an optimal second advisory for simple reconfiguration cost-dominated problem instances. ∎

2. Simple Static Cost-Dominated Problem Instances with $d > 1$

Definition 3 (static cost-dominated M-ε-d-CSAs problem instance) *Static cost-dominated M-ε-d-CSAs problem instances are those in which $\beta^R = 0$.*

An important characteristic of static cost-dominated instances is that if there is a static cost benefit in a single time step of using one configuration over another, the reconfiguration cost required to achieve this cost benefit by reconfiguring is never prohibitive. In these instances, it is always best to use a configuration with the minimum static cost at each time step.

Lemma 2 *The single optimal first advisory for simple static cost-dominated problem instances uses the unique configuration with the lowest static cost at each time step $k \in \{1, \ldots, K\}$.*

Proof of Lemma 2 The definition of simple instances ensures that the optimal first advisory is unique. The definition of static cost-dominated instances ensures that lower advisory costs can always be achieved by reconfiguring as much as is required to achieve the lowest static cost in each next time step. Taken together, these definitions imply that there must be a unique configuration achieving the lowest static cost in each time step, and the unique optimal first advisory uses this configuration at each time step. ∎

Proposition 2 *If $d > 1$, FBVISAS does not return a second advisory for simple static cost-dominated problem instances.*

Proof of Proposition 2 Consider partial advisories to any configuration C at any time step k from the initial configuration at $k = 0$. For simple static cost-dominated instances, the CSA-minimum cost partial advisory will use the unique static cost-minimizing configuration from time steps 1 to $k - 1$.

Consider partial advisories from any configuration C at any time step k to time step K. For simple static cost-dominated instances, the CSA-minimum cost partial advisory will use the unique static cost-minimizing configuration from time steps $k + 1$ to K.

Two such corresponding partial advisories are combined into a single advisory that is evaluated by FBVISAS when it investigates any configuration C at any time step k. Such investigated advisories never achieve a difference from the optimal first advisory of more than 1 because they only diverge from the optimal advisory at time step k. Therefore, the potential second advisories investigated by FBVISAS all fail to meet the difference constraint (10) and so no second advisory is returned. ∎

Appendix E: Forward A^* Algorithm

The Forward A^* algorithm (ForwardA^*) is specified in Algorithm 6.

Algorithm 6 Forward$A^*(\mathcal{C}, T)$

Require: $\mathcal{C} = \{\mathcal{C}_k\}_{k=0}^{K}$ {Valid configuration schedule advisories}

Require: T {Traffic situation data}

  `closed` $\leftarrow \emptyset$

  `open` $\leftarrow$ priority queue containing C_0 with key 0

  $C_k \leftarrow$ minimum-key configuration in `open`

  while $C_k \notin \mathcal{C}_K$ **do**

    Add C_k to `closed`

    for $C_{k+1} \in \mathcal{C}_{k+1}$ **do**

      $\underline{J} \leftarrow \underline{J}_k(C_k) + g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1})$

      if $C_{k+1} \in$ `open` and $\underline{J} < \underline{J}_{k+1}(C_{k+1})$ **then**

        Remove C_{k+1} from `open`

      if $C_{k+1} \notin$ `open` **then**

        $\underline{J}_{k+1}(C_{k+1}) \leftarrow \underline{J}$

        Add C_{k+1} to `open` with key $\underline{J}_{k+1}(C_{k+1}) + \hat{J}_{k+1}(C_{k+1})$

        $\underline{C}_k^{\dagger}(C_{k+1}) \leftarrow C_k$

    $C_k \leftarrow$ minimum-key configuration in `open`

  Construct C from C_k by iteratively using $\underline{C}_{k-1}^{\dagger}(C_k)$

  return $C, \{\underline{J}_k(C_k)\}_{k=1}^{K}, \{\underline{C}_{k-1}^{\dagger}(C_k)\}_{k=1}^{K}$

Appendix F: Forward Distinct A^* Algorithm

The Forward Distinct A^* algorithm (FDA^*) is specified in Algorithm 7. This algorithm specification uses $\phi_{\max}$, which is the maximum possible configuration difference:

$$\phi_{\max} = \max_{k \in \{1,2,\ldots,K\}} \max_{C_k, C'_k \in \mathcal{C}_k} \phi(C_k, C'_k).$$

For the configuration difference metric (12) used in this article, $\phi_{\max} = 1$.

Appendix G: Proofs and Discussion of Properties of the Forward Distinct A^* Algorithm

Definition 4 (d-second-CSA problem) *Consider M-ε-d-CSAs instances in which $M = 2$, $\varepsilon = \infty$, and there is a unique optimal first advisory ($|\mathcal{C}^\star_{CSA}(\mathcal{C}, T)| = 1$). For such instances, finding a second advisory involves solving*

$$\begin{aligned} &\textit{minimize } g(C^2, T) \\ &\textit{subject to } C^2_k \in \mathcal{C}_k, \quad k = 0, 1, 2, \ldots, K \\ &\Phi(C^1, C^2) \geq d, \end{aligned}$$

which will be referred to as the d-second-CSA problem.

Lemma 3 *If $\lambda \geq 0$ and $\hat{J}_k(C_k)$ is an underestimate of the remaining cost $\sum_{k'=k+1}^{K} g_{k'}(C_{k'-1}, T_{k'-1}, C_{k'}, T_{k'})$ of a partial advisory starting from C_k, then the advisory returned by FDA^* minimizes the Lagrangian of the d-second-CSA problem.*

Proof of Lemma 3 The FDA^* algorithm is the forward A^* algorithm, but with an *advisory rank* function $\underline{R}_k$ used instead of the cost $\underline{J}_k$. The advisory rank function can be computed as a sum of the *configuration rank* r_k over the configurations in an advisory or partial advisory. This configuration rank function involves both the cost g_k as well as the configuration difference between C_k and C^1_k:

$$\begin{aligned} \underline{R}_k(\{C_{k'}\}_{k'=0}^{k}, \{C^1_{k'}\}_{k'=0}^{k}, T, \lambda) &= \sum_{k'=1}^{k} r_{k'}(C_{k'-1}, C_{k'}, C^1_{k'}, T_{k-1}, T_k, \lambda) \\ &= \sum_{k'=1}^{k} \left[g_{k'}(C_{k'-1}, T_{k'-1}, C_{k'}, T_{k'}) + \lambda(\phi_{\max} - \phi(C_{k'}, C^1_{k'})) \right]. \end{aligned}$$

The contribution to the rank from each configuration in the advisory is nonnegative when $\lambda \geq 0$ because $g_k(\cdot, \cdot, \cdot, \cdot) \geq 0$ and because $\phi_{\max} \geq \phi(\cdot, \cdot)$. Since FD$A^*$ is simply the A^* algorithm with a

Algorithm 7 FD$A^*(\mathcal{C}, T, J^\star, d, C^1, \lambda)$

Require: $\mathcal{C} = \{\mathcal{C}_k\}_{k=0}^K$ {Valid configuration schedule advisories}

Require: T {Traffic situation data}

Require: $J^\star$ {Minimum cost for corresponding CSA problem instance}

Require: d {Parameter from M-ε-d-CSAs problem instance specification}

Require: C^1 {First advisory for M-ε-d-CSAs problem instance}

Require: λ {Algorithm parameter}

```
closed ← ∅
open ← priority queue containing C_0 with key 0
C_k ← minimum-key configuration in open
while C_k ∉ C_K do
    Add C_k to closed
    for C_{k+1} ∈ C_{k+1} do
        J ← J_k(C_k) + g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1})
        P^1 ← P^1_k(C_k) + φ(C_{k+1}, C^1_{k+1})
        R ← J + λ((k+1)φ_max − P^1)
        if C_{k+1} ∈ open and R ≤ R(C_{k+1}) then
            Remove C_{k+1} from open
        if C_{k+1} ∉ open then
            R_{k+1}(C_{k+1}) ← R
            J_{k+1}(C_{k+1}) ← J
            P^1_{k+1}(C_{k+1}) ← P^1
            C^†_k(C_{k+1}) ← C_k
            R(C_{k+1}) ← R_{k+1}(C_{k+1}) + Ĵ_{k+1}(C_{k+1})
            Add C_{k+1} to open with key R(C_{k+1})
    C_k ← minimum-key configuration in open
Construct C^2 from C_k by iteratively using C^†_{k−1}(C_k)
return C^2
```

nonnegative configuration rank as the configuration cost, if FDA^* uses an underestimate of the rank that will be incurred by the remainder of an advisory starting at C_{k+1} when computing $R(C_{k+1})$, then it will return a second advisory that minimizes the sum of the configuration rank. This is indeed

the case when $\hat{J}_k(C_k)$ is an underestimate, as assumed. Let $C^{2,\star}$ be the advisory rank-minimizing advisory returned by FDA^*.

Next, consider a modified configuration rank function

$$r'_k(C_{k-1}, C_k, C^1_k, T_{k-1}, T_k, \lambda) = r_k(C_{k-1}, C_k, C^1_k, T_{k-1}, T_k, \lambda) - \lambda\phi_{\max} + \lambda\frac{d}{K}.$$

Since the additional terms are both constants, $C^{2,\star}$ minimizes the sum of the modified rank as well. An A^*-based algorithm cannot be used directly to minimize this modified rank because it can be negative, which is why FDA^* minimizes the unmodified rank, which is nonnegative. We know that $C^{2,\star}$ will minimize the both the sum of the unmodified rank and the sum of the modified rank because each advisory has the same number of configurations. The sum of the modified rank of the configurations in an advisory is

$$\begin{aligned}
\sum_{k=1}^{K} r'_k(C_{k-1}, C_k, C^1_k, T_{k-1}, T_k, \lambda) &= \sum_{k=1}^{K}\left[r_k(C_{k-1}, C_k, C^1_k, T_{k-1}, T_k, \lambda) - \lambda\phi_{\max} + \lambda\frac{d}{K}\right] \\
&= \sum_{k=1}^{K}\left[g_k(C_{k-1}, T_{k-1}, C_k, T_k) + \lambda(\phi_{\max} - \phi(C_k, C^1_k)) - \lambda\phi_{\max} + \lambda\frac{d}{K}\right] \\
&= \sum_{k=1}^{K}\left[g_k(C_{k-1}, T_{k-1}, C_k, T_k) - \lambda\phi(C_k, C^1_k) + \lambda\frac{d}{K}\right] \\
&= g(C^2, T) - \lambda\Phi(C^2, C^1) + \lambda d \\
&= g(C^2, T) + \lambda(d - \Phi(C^1, C^2)). \qquad \text{(G1)}
\end{aligned}$$

For any $C^2 \in \mathcal{C}$ and $\lambda \geq 0$, we can specify the Lagrangian $L(C^2, \lambda)$ for the d-second-CSA problem as

$$L(C^2, \lambda) = g(C^2, T) + \lambda(d - \Phi(C^1, C^2)).$$

This is identical to (G1), so FDA^* finds a $C^2 \in \mathcal{C}$ that minimizes the Lagrangian $L(C^2, \lambda)$ of the d-second-CSA problem. ∎

Lemma 3 means that FDA^* implements the Lagrange dual function $h(\lambda) = \min_{C^2 \in \mathcal{C}} L(C^2, \lambda)$ for the d-second-CSA problem. By weak duality we know that $h(\lambda)$ provides a lower bound on the minimum cost for the primal d-second-CSAs problem for any $\lambda \geq 0$.

This Lemma is possible because the assumptions that $|\mathcal{C}_{\text{CSA}}(\mathcal{C}, T)| = 1$ and $M = 2$ eliminate the combinatorial nature of the M-ε-d-CSAs problem. The difference function (11) decomposes as

a sum over the differences between the two advisories at each time step, so the A^* algorithm still minimize the cost after the difference constraint is incorporated into it (see Appendix E). In fact, the d-second-CSAs problem is a type of constrained shortest path problem. Related algorithms also leverage Lagrange duality to solve constrained shortest path problems [31, 32].

Proposition 3 *Suppose $M = 2$, $\varepsilon = \infty$, there is a unique optimal first advisory ($|\mathcal{C}_{CSA}(\mathcal{C}, T)| = 1$), and $\hat{J}_k(C_k)$ is an underestimate of the remaining cost. If $\lambda^\star$ is optimal for the Lagrange dual problem of the d-second-CSA problem (i.e., it maximizes $h(\lambda)$ over all $\lambda \geq 0$) and strong duality is satisfied, then the FDA* algorithm with $\lambda = \lambda^\star$ returns an advisory that achieves the same Lagrangian value as any second advisory that is optimal for the d-second-CSA problem.*

Proof of Proposition 3 Let $C^{2,\star}$ be any second advisory that is optimal for the d-second-CSA problem. Strong duality is satisfied (by assumption), which means that $g(C^{2,\star}, T) = h(\lambda^\star)$ and that $C^{2,\star}$ minimizes $L(C^2, \lambda^\star)$. By Lemma 3, we know that FDA^* also returns a second advisory that minimizes $L(C^2, \lambda)$. Therefore, when $\lambda = \lambda^\star$, the advisory returned by FDA^* and $C^{2,\star}$ must achieve the same minimum Lagrangian value. ∎

This result can be extended to a problem like the d-second-CSA problem but with $\varepsilon \in [0, \infty)$. This is referred to as the ε-d-second-CSA problem.

Definition 5 (ε-d-second-CSA problem) *Consider M-ε-d-CSAs instances in which $M = 2$, $\varepsilon \in [0, \infty)$, and there is a unique optimal first advisory ($|\mathcal{C}^\star_{CSA}(\mathcal{C}, T)| = 1$). For such instances, finding a second advisory involves solving*

$$\begin{aligned}
&\textit{minimize } \ g(C^2, T) \\
&\textit{subject to } C^2_k \in \mathcal{C}_k, \quad k = 0, 1, 2, \ldots, K \\
&\qquad \frac{g(C^2, T) - g(C^1, T)}{g(C^1, T)} \leq \varepsilon, \qquad \text{(G2)} \\
&\qquad \Phi(C^1, C^2) \geq d, \qquad \text{(G3)}
\end{aligned}$$

which will be referred to as the ε-d-second-CSA problem.

Let λ_1 be the dual variable corresponding to constraint (G2) and λ_2 be the dual variable corresponding to constraint (G3).

Corollary 1 *Suppose $M = 2$, there is a unique optimal first advisory ($|\mathcal{C}_{CSA}(\mathcal{C}, T)| = 1$), and $\hat{J}_k(C_k)$ is an underestimate of the remaining cost,. If $\lambda^\star = [\lambda_1^\star, \lambda_2^\star]$ is optimal for the Lagrange dual problem of the ε-d-second-CSA problem and strong duality is satisfied, then the FDA* algorithm with $\lambda = \frac{\lambda_2^\star}{1+\lambda_1^\star}$ returns an advisory that achieves the same Lagrangian value as any second advisory that is optimal for the ε-d-second-CSA problem.*

Proof of Corollary 1 The Lagrangian for the ε-d-second-CSA problem is

$$\begin{aligned} L(C^2, \lambda) &= g(C^2, T) + \lambda_1(g(C^2, T) - (1+\varepsilon)J^\star) + \lambda_2(d - \Phi(C^1, C^2)) \\ &= (1+\lambda_1)g(C^2, T) - \lambda_1(1+\varepsilon)J^\star + \lambda_2(d - \Phi(C^1, C^2)). \end{aligned} \tag{G4}$$

The middle term is not impacted by C^2. Furthermore, since $1 + \lambda_1 \geq 0$, minimizing the Lagrangian (G4) over $C^2 \in \mathcal{C}$ is the same as finding the advisory that minimizes

$$g(C^2, T) + \frac{\lambda_2}{1+\lambda_1}(d - \Phi(C^1, C^2)).$$

When $\lambda = \frac{\lambda_2}{1+\lambda_1}$, this is the cost function for which FDA^* finds an optimal advisory. The result can be proven by using the techniques used in the proofs of Lemma 3 and Proposition 3. ∎

These results motivate the FDA^* algorithm because under the right circumstances, it returns a second advisory that satisfies a condition that is necessarily satisfied by optimal second advisories. Aside from the insufficiency of this condition, there are some important qualifications to this motivation. The M-ε-d-CSAs problem instances that give rise to the d-second-CSA and ε-d-second-CSA problems are only a subset of all M-ε-d-CSAs problem instances. Also, we also have no reason to believe that strong duality will hold for second-CSA problem instances. For cases when strong duality does not hold, algorithms developed for constrained shortest path problems could be deployed to find lower-cost second advisories [32]. Given that we are interested in quickly finding advisories for instances in which $M > 2$, we have not pursued such approaches. Finally, Proposition 3 and Corollary 1 require the value of $\lambda^\star$ or $\frac{\lambda_2^\star}{1+\lambda_1^\star}$, respectively. For the d-second-CSA problem, $\lambda^\star$ can be found because an implication of Lemma 3 is that by searching over $\lambda \geq 0$ and repeatedly calling

FDA^*, we can solve the dual of the d-second-CSA problem:

$$\text{maximize } h(\lambda)$$

$$\text{subject to } \lambda \geq 0.$$

Since $h(\lambda)$ is concave and being maximized over a convex set, this is a convex problem with a single real-valued variable. It could be solved with a simple bisection search in which FDA^* is called at each candidate λ value to determine the value of $h(\lambda)$. Finding the optimal dual variable in this way is not pursued in this research because such repeated calls of the FDA^* algorithm are computationally prohibitive for our application and because we are interested in problem instances for which $M > 2$. These considerations led to the development of the FDA^*-SC algorithm.

Appendix H: Forward Distinct A^* with Shortcuts Algorithm

The Forward Distinct A^* with Shortcuts algorithm (FDA^*-SC) is specified in Algorithm 8. For FDA^*-SC, the `open` priority queue functionality must be adjusted. The queue key is set up such that any configuration labeled as a shortcut is lower than any configuration not labeled as a shortcut. Among shortcut configurations or non-shortcut configurations, the key works as usual (lower keys given higher priority than higher keys). Therefore, if shortcut configurations are in the queue, then the minimum-key configuration is the shortcut configuration with the lowest key.

If $\hat{P}^m(C_k)$ is always an underestimate of the difference-to-go, then the algorithm will not incorrectly try to take a shortcut with partial advisories that will never become distinct enough from advisory m. In this article, we define $\hat{P}^m(C_k) \triangleq 0$ for all m and C_k to ensure that this is an underestimate. The upper difference bound $\check{P}^m(C_k)$ ensures that only configurations that can possibly achieve sufficient distinctness are added to the `open` queue. For the difference metric (11) used in this article, $\check{P}^m(C_k) = K - k$ works because at the most one unit of difference can be earned in each remaining time step. Finally, $\Phi_{\max}$ is the maximum possible advisory difference metric value.

Acknowledgments

We are grateful to Karl Bilimoria and Michael Drew for their helpful suggestions regarding the problem defined in this article. We thank Chok Fung (Jack) Lai for his patient assistance with

Algorithm 8 FDA^*-SC$(\mathcal{C}, T, J^\star, \varepsilon, d, \mathcal{C}^M, \lambda, \{\bar{J}_k(C_k)\}_{k=0}^{K-1}, \{\bar{C}_{k+1}^\dagger(C_k)\}_{k=0}^{K-1}, \varepsilon')$

Require: $\mathcal{C} = \{\mathcal{C}_k\}_{k=0}^K$, T {Constraints and traffic data from corresponding CSA instance}

Require: $J^\star$ {Minimum cost for corresponding CSA problem instance}

Require: ε, d {Parameters in M-ε-d-CSAs problem instance specification}

Require: $\mathcal{C}^M$ {Advisories found so far for M-ε-d-CSAs problem instance}

Require: λ, ε' {Algorithm parameters}

Require: $\{\bar{J}_k(C_k)\}_{k=0}^{K-1}$, $\{\bar{C}_{k+1}^\dagger(C_k)\}_{k=0}^{K-1}$ {Partial advisory specifications and costs}

$m \leftarrow |\mathcal{C}^M| + 1$

`closed` $\leftarrow \emptyset$

`open` $\leftarrow$ priority queue containing C_0 with key 0

$C_k \leftarrow$ minimum-key configuration in `open`

while not ($C_k \in \mathcal{C}_K$ **and** $\frac{\underline{J}_K(C_K) - J^\star}{J^\star} \leq \varepsilon$ **and** $\min_{m' \in 1,\ldots,m-1} \underline{P}_K^{m'}(C_K) \geq d$) **and** C_k not a shortcut **do**

Add C_k to `closed`

for $C_{k+1} \in \mathcal{C}_{k+1}$ **do**

$\underline{J} \leftarrow \underline{J}_k(C_k) + g_{k+1}(C_k, T_k, C_{k+1}, T_{k+1})$

for $m' = 1, \ldots, m-1$ **do**

$\underline{P}^{m'} \leftarrow \underline{P}_k^{m'}(C_k) + \phi(C_{k+1}, C_{k+1}^{m'})$

if $\frac{\underline{J} + \hat{J}_{k+1}(C_{k+1}) - J^\star}{J^\star} \leq \varepsilon$ **and** $\min_{m' \in \{1,\ldots,m-1\}} \left[\underline{P}^{m'} + \check{P}^{m'}(C_{k+1})\right] \geq d$ **then**

$\underline{R} \leftarrow \frac{\underline{J}}{J^\star} + \lambda \frac{1}{m-1} \sum_{m'=1}^{m-1} \frac{(k+1)\phi_{\max} - \underline{P}^{m'}}{\Phi_{\max} - d + 1}$

if $C_{k+1} \in$ `open` and $\underline{R} < \underline{R}_{k+1}(C_{k+1})$ **then**

Remove C_{k+1} from `open`

if $C_{k+1} \notin$ `open` **then**

$\underline{R}_{k+1}(C_{k+1}) \leftarrow \underline{R}$

$\underline{J}_{k+1}(C_{k+1}) \leftarrow \underline{J}$

for $m' = 1, \ldots, m-1$ **do**

$\underline{P}_{k+1}^{m'}(C_{k+1}) \leftarrow \underline{P}^{m'}$

$\underline{C}_k^\dagger(C_{k+1}) \leftarrow C_k$

$R(C_{k+1}) \leftarrow \underline{R}_{k+1}(C_{k+1}) + \frac{\hat{J}_{k+1}(C_{k+1})}{J^\star}$

Add C_{k+1} to `open` with key $R(C_{k+1})$

if $\frac{\underline{J}(C_{k+1}) + \bar{J}_{k+1}(C_{k+1}) - J^\star}{J^\star} \leq \varepsilon'$ **and** $\min_{m' \in \{1,\ldots,m-1\}} [\underline{P}_{k+1}^{m'}(C_{k+1}) + \hat{P}_{k+1}^{m'}(C_{k+1})] \geq d$ **then**

Label C_{k+1} as a shortcut and add it to `open` with key $\underline{J}(C_{k+1}) + \bar{J}_{k+1}(C_{k+1})$

$C_k \leftarrow$ minimum-key configuration in `open`

Construct C^m from C_k by iteratively using $\underline{C}_{k-1}^\dagger(C_k)$ and $\bar{C}_{k+1}^\dagger(C_k)$; add C^m to $\mathcal{C}^M$

return $\mathcal{C}^M$

our software implementation of the algorithms described in this article. The historical airspace configuration data used in this research was provided by ATAC.

References

[1] Federal Aviation Administration, "Order JO 7210.3X Facility Operation and Administration," http://www.faa.gov/air_traffic/publications/atpubs/FAC/index.htm, February 2012.

[2] Cano, M., Sánchez-Escalonilla, P., and Dorado, M. M., "Complexity Analysis in the Next Generation of Air Traffic Management System," *Proc. of AIAA/IEEE Digital Avionics Systems Conference*, October 2007.

[3] Bloem, M. and Kopardekar, P., "Combining Airspace Sectors for the Efficient Use of Air Traffic Control Resources," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, HI, August 2008.

[4] Bloem, M., Gupta, P., and Kopardekar, P., "Algorithms for Combining Airspace Sectors," *Air Traffic Control Quarterly*, Vol. 17, No. 3, 2009, pp. 245–268.

[5] Drew, M. C., "A Method of Optimally Combining Sectors," *AIAA Aviation Technology, Integration and Operations Conference*, Hilton Head, SC, September 2009.

[6] Bloem, M. and Gupta, P., "Configuring Airspace Sectors with Approximate Dynamic Programming," *Proc. of International Congress of the Aeronautical Sciences*, Nice, France, September 2010.

[7] Tien, S.-L., Hoffman, R., and Schonfeld, P., "En Route Sector Combination Scheme to Minimize Air Traffic Controller Staffing," *Proc. of Transportation Res. Board Annual Meeting*, Washington, DC, January 2012.

[8] Bloem, M., Drew, M. C., Lai, C. F., and Bilimoria, K., "Advisory Algorithm for Scheduling Open Sectors, Operating Positions, and Workstations," *AIAA Aviation Technology, Integration, and Operations Conference*, Indianapolis, IN, September 2012.

[9] Bloem, M., Drew, M., Lai, C. F., and Bilimoria, K., "Advisory Algorithm for Scheduling Open Sectors, Operating Positions, and Workstations," *AIAA Journal of Guidance, Control, and Dynamics*, 2014.

[10] Lee, P. U., Mogford, R., Bridges, W., Buckley, N., Evans, M., Gujral, V., Lee, H., Peknik, D., and Preston, W., "An Evaluation of Operational Airspace Sectorization Integrated System (OASIS) Advisory Tool," *AIAA Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA, August 2013.

[11] Nguyen, T., Do, M., Gerevini, A. E., Serina, I., Srivastava, B., and Kambhampati, S., "Generating Diverse Plans to Handle Unknown and Partially Known User Preferences," *Artificial Intelligence*, Vol. 190, October 2012, pp. 1–31.

[12] Bloem, M. and Bambos, N., "An Approach for Finding Multiple Area of Specialization Configuration Advisories," *AIAA Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA,

August 2013.

[13] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions of Systems Science and Cybernetics*, Vol. 4, No. 2, July 1968, pp. 100–107.

[14] Byers, T. H. and Waterman, M. S., "Determining All Optimal and Near-Optimal Solutions When Solving Shortest Path Problems by Dynamic Programming," *Operations Research*, Vol. 32, No. 6, 1984, pp. 1381–1384.

[15] Trevisan, L., "Inapproximability of Combinatorial Optimization Problems," Technical Report TR04–065, Electronic Colloquium on Computational Complexity, July 2004.

[16] Kleinberg, J. and Éva Tardos, *Algorithm Design*, Addison Wesley, 2005.

[17] Bellman, R. and Kalaba, R., "On kth Best Policies," *Journal of the SIAM*, Vol. 8, No. 4, December 1960, pp. 582–588.

[18] Hoffman, W. and Pavley, R., "A Method for the Solution of the Nth Best Path Problem," *Journal of the ACM*, Vol. 6, No. 4, October 1959, pp. 506–514.

[19] Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms," Memorandum RM-5433-1-PR, United States Air Force Project RAND, September 1963.

[20] Eppstein, D., "Finding the k Shortest Paths," *SIAM Journal of Computing*, Vol. 28, No. 2, 1998, pp. 652–673.

[21] Martins, E., Pascoal, M., and dos Santos, J., "Deviation Algorithms for Ranking Shortest Paths," *International Journal of Foundations of Computer Science*, Vol. 10, No. 3, 1999, pp. 247–262.

[22] Jiménez, V. M. and Marzal, A., "Computing the K Shortest Paths: A New Algorithm and an Experimental Comparison," *Algorithm Engineering*, edited by J. Vitter and C. Zaroliagis, Vol. 1668 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 1999, pp. 15–29.

[23] Suurballe, J. W., "Disjoint Paths in a Network," *Networks*, Vol. 4, 1974, pp. 125–145.

[24] Suurballe, J. W. and Tarjan, R. E., "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, Vol. 14, 1984, pp. 325–336.

[25] Andreas, A. K. and Smith, J. C., "Exact Algorithms for Robust k-Path Routing Problems," *Proc. of the GO*, 2005.

[26] Peng, C. and Shen, H., "An Improved Approximation Algorithm for Computing Disjoint QoS-Paths," *Proc. of IEEE International Conference on Systems and Networking*, April 2006.

[27] Andreas, A. K., Smith, J. C., and Küçükyavuz, S., "Brand-and-Price-and-Cut Algorithms for Solving the Reliable h-Paths Problem," *Journal of Global Optimization*, Vol. 42, No. 4, December 2008.

[28] Loh, R. C., Soh, S., and Lazarescu, M., "Edge Disjoint Paths with Minimum Delay Subject to Reliability Constraint," *Proc. of IEEE Asia-Pacific Conference on Communications*, Shanghai, China, October 2009.

[29] Loh, R.-C., Soh, S., and Lazarescu, M., "An Approach to Find Maximal Disjoint Paths with Reliability and Delay Constraints," *Proc. of IEEE International Conference on Advanced Information Networking and Applications*, June 2009.

[30] Yang, Y., Wang, S., Hu, X., Li, J., and Xu, B., "A Modified K-Shortest Paths Algorithm for Solving the Earliest Arrival Problem on the Time-Dependent Model of Transportation Systems," *Proc. of International MultiConference of Engineers and Computer Scientists*, Hong Kong, China, March 2012.

[31] Handler, G. Y. and Zang, I., "A Dual Algorithm for the Constrained Shortest Path Problem," *Networks*, Vol. 10, 1980, pp. 293–310.

[32] Carlyle, W. M., Royset, J. O., and Wood, R. K., "Lagrangian Relaxation and Enumeration for Solving Constrained Shortest-Path Problems," *Networks*, Vol. 52, No. 4, December 2008, pp. 256–270.